1-1-2010

Design of Regular Reversible Quantum Circuits

Dipal Shah
Portland State University

10.15760/etd.129

Design of Regular Reversible Quantum Circuits

by

Dipal Shah

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

Dissertation Committee:
Marek A. Perkowski, Chair
Garrison Greenwood
Xiaoyu Song
Cynthia Brown
Laszlo Csanky

Portland State University

ABSTRACT

The computing power in terms of speed and capacity of today's digital computers has improved tremendously in the last decade. This improvement came mainly due to a revolution in manufacturing technology by developing the ability to manufacture smaller devices and by integrating more devices on a single die. Further development of the current technology will be restricted by physical limits since it won't be possible to shrink devices beyond a certain size. Eventually, classical electrical circuits will encounter the barrier of quantum mechanics. The laws of quantum mechanics can be used for building computing systems that work on the principles of quantum mechanics. Thus quantum computing has drawn the interest of many top scientists in the world. Ion Trap technology is one of the most promising prospective technologies for building quantum computers. This technology allows the placement of qubits - ions in 1-, 2- and 3-dimensional regular structures. Development of efficient algorithms and methodologies for designing reversible quantum circuits is one of the most rapidly growing areas of research. All existing algorithms for synthesizing quantum circuits use multi-input Toffoli gates that have very high quantum cost in terms of electromagnetic pulses. They also do not use the opportunity of regular structures provided by the Ion Trap technology.

In this thesis I present a completely new methodology for synthesizing quantum circuits that use only small (3×3) Toffoli gates and new gate families that have similar properties

and use regular structures. These methods are for both binary and multiple valued quantum circuits. All my methods require adding some limited number of ancilla qudits but dramatically decrease the quantum cost of the synthesized circuits. I also present a new family of gates called "D-gates" that allows synthesis of quantum and reversible logic functions using structures called layered diagrams. The designed circuits can be directly mapped to a Quantum Logic Array implemented using the Ion Trap technology.

DEDICATION

This Dissertation is dedicated to:

My parents Sunila and Arvind Shah;

My wife Swati Shah;

My son Ishan Shah;

My Aunt Sudha Desai;

ACKNOWLEDGMENTS

I am deeply grateful to my dissertation advisor, Professor Marek A. Perkowski, who allowed me the opportunity to take the next step in my scientific vocation and do my PhD. He allowed me the scope to follow my thoughts and develop my scientific skills, whilst always being available to provide guidance, inspiration and discussion whenever it was needed over a period of several years. I would like to thank Professor Xiaoyu Song and Professor Garrison Greenwood for serving on both of my dissertation proposal defense and comprehensive exam committee. I would like to extend thanks to Professor Cynthia Brown, Professor Xiaoyu Song, Professor Garrison Greenwood and Professor Laszlo Csanky for being on my dissertation committee. I am very thankful to Professor Cynthia Brown for her enormous suggestions in improving quality of my dissertation. I would like to thank my friends for their support and inspiration. Finally, I would like to express my gratitude to my parents, my wife Swati, son Ishan for their love, understanding, patience and support to sustain me through the end of my PhD.

TABLE OF CONTENTS

	Page
Abstract	**i**
Dedication	**iii**
Acknowledgements	**iv**
List of Tables	**x**
List of Figures	**xi**
1. Introduction.	**1**
2. Binary, Ternary and Hybrid (mixed Binary-Ternary) Quantum Gates.	**15**
2.1. Introduction.	15
2.1.1. Overview of Ion Trap technology for Quantum Computing.	15
2.1.2. Why reversible logic circuits?	17
2.1.3. How reversible logic applies to quantum circuits?	18
2.1.4. What are oracles and why are they important in quantum computing?	19
2.1.5. Are multiple-level circuits realizable in quantum technologies, which ones?	20
2.1.6. Practical examples of multiple-valued quantum circuits.	22
2.1.7. Costs of gates and circuits in multiple-valued quantum technologies.	22
2.2. Ion Trap technology for quantum computers.	24
2.2.1. Ion trapping.	24
2.2.2. Loading Ions.	29
2.2.3. Cadmium 111 as a Qubit.	30
2.2.4. Initializing and detecting the Qubit states.	31
2.2.5. Two ions entangling gate.	33

2.2.6. Quantum charge coupled device (QCCD). 34
2.2.7. Quantum Logic Array (QLA) Architecture. 37
2.3. Mathematical Background on Binary Quantum Circuits, qubits and measurements. 39
2.3.1. Bloch Sphere, rotations and gates. 39
2.3.2. Basic operations of Square-Root-Of-NOT and its adjoint gates. 48
2.4. Mathematical Background on Ternary Quantum Logic. 50
2. 4.1. Qudits and Dirac Notation for ternary quantum circuits. 50
2.4.2. Potentially useful Operators. 52
2.4.3. Galois Field 3 Logic. 53
2.4.4. Ternary 1-qutrit permutative gates. 53
2.4.5. Chrestenson gate as a generalization of Hadamard with different properties. 55
2.5. Two-Qutrit gates. 57
2.5.1. Ternary Muthukrishnan-Stroud gates. 57
2.5.2. Ternary Feynman gate. 58
2.6. Ternary gates with 3 or more qutrits. 60
2.6.1. Ternary Toffoli gates. 60
2.6.2. More 3-qutrit gates. 63
2.6.3. Realization of ternary Sum of Minterms using generalized Toffoli gates. 64
2.6.4. Why and when multiple-valued quantum circuits can be used in quantum computing? 66
2.7. Background on hybrid binary-ternary quantum circuit. 68
2.8. Conclusions and related work in the thesis. 70

3. Binary Tree and Decision Diagram Expansions for Quantum Circuits Synthesis. 74

3.1. Types of Logic. 74

3.2. Binary Reed-Muller Logic. 80

4. Towards Efficient Computer Aided Design Methods for D-level Quantum Computers. 89

4.1. Towards Computer Aided Design of d-level Quantum Computers. 89

4.2. How to synthesize quantum circuits on level of pulses, simple gates and circuit blocks? 90

4.3. Reed-Muller Logic, Galois Logic and Quantum Computing. 93

4.4. Highly Testable Quantum Circuits. 97

4.5. From binary to multiple-level Quantum Circuits. 100

4.6. Some types of multiple-valued Quantum Circuits. 102

4.7. Heuristic search methods to Synthesize Quantum Circuits from the above-introduced gates and circuits. 109

4.8. Heuristic Search versus Nature-inspired Evolutionary Search. 112

4.9. Introductory illustration of some basic ideas of this dissertation. 114

4.10. Conclusions. 126

5. Regularity and Ancilla Bits in Binary Quantum Circuits. 128

5.1. Some preliminary ideas. 128

5.2. From binary function expansions to trees and lattices. 129

5.2.1. Combining binary trees and binary lattices in regular layouts. 129

5.2.2. Relations between logic expansions of Boolean Functions and regular diagrams. 134

5.3. Characterization of basic regular structures. 139

5.4. The main principle of forward and inverse expansions. 145

5.5. Binary Orthogonal Regular Diagrams. 151

6. From Kronecker Functional Decision Diagram to Regular Quantum Array of Toffoli Gates. 157

6.1. Representation of Positive Davio gate as a Toffoli gate and Invention of the d-gate. 157

6.2. Kronecker Functional Decision Diagram. 158

6.3. Definition of Lattice Diagrams. 159

6.4. Creating Positive Davio lattice. 162

6.4.1. Algorithm pseudo code for creating Lattice Diagrams. 163

6.5. Description of the Algorithm (KFDD to QA with Toffoli gates). 165

6.6. Quantum Array Optimization by Creating Efficient KFDD. 171

6.7. Experimental Results. 174

7. Development of the Dipal Gate Family and Design of the Layered Diagrams. 176

7.1. Layered-based Regular Structures with Shannon Expansion. 176

7.2. The concept and the design of Dipal gate. 179

7.3. Layered-based expansions with Toffoli gate and Dipal Gate Family. 183

7.4. Algorithm pseudo code for creating the Shannon Lattice Diagrams. 187

7.5. Experimental Results. 189

8. From Binary to Multiple Valued Quantum Array. 190

8.1. Other Structures for Layered-based expansions. 190

8.2. Generalizations from binary to ternary. 192

9. Galois Field Logic, Group Logic and Their use for Regular Quantum Circuit Synthesis. **197**

9.1. Binary Generalized Reed-Muller Forms and Classical Binary AND-EXOR Hierarchy. 198

9.2. Binary ESOP Logic. 201

9.3. Galois Field Logic. 204

9.3.1. Definitions, Literals and basic concepts. 204

9.3.2. Detailed analysis of binary expansions as our research base. 211

9.3.3. Derivation of Davio Expansions for the ternary case. 215

9.3.4. Ternary Pseudo-Kronecker Expansions. 224

9.3.5. Ternary Lattices. 235

9.4. The GF-Kronecker type Expansions. 236

9.5. The GF Hierarchy. 239

9.6. Quantum Circuits based on Galois Fields. 241

9.7. MVL Orthogonal and Linearly Independent Expansions. 243

9.8. Orthogonal Expansion Structures for Quaternary Logic. 244

9.9. Reverse Expansions of Galois Field type. 245

10. Conclusions and future work. **250**

10.1. Key accomplishments of my research. 250

10.3 Future work. 257

11. References **262**

LIST OF TABLES

Table **Page**

Table 2.1 Unitary matrix equations for P, X, Y and Z rotations. 42

Table 2.2 Galois Field 3 operations. 53

Table 2.3 1-qutrit ternary permutative transforms. 54

Table 2.4 1-qudit ternary gates and their inverse gates. 55

Table 2.5 Ternary Truth table of a half-adder. 65

Table 2.6 Truth table of a mixed binary-ternary gate. The qudit x_1 is binary, qudits x_2 and x_3 are ternary. 69

Table 6.1 Experimental results in terms of number of gates in the synthesized circuits as well as respective quantum cost. 175

Table 7.1 shows experimental results for synthesized circuits using d-gate family. 189

Table 9.1.1 Classical, Reed-Muller binary trees, diagrams and expansions. 203

LIST OF FIGURES

Figure **Page**

Figure 1.1 Organization of dissertation. 14

Figure 2.1 Figure 2.1 Schematic diagram of an ion trap electrode. (a) Asymmetric quadrupole trap with a 400 μm diameter ring electrode and a 300 μm gap between the form electrodes. The rf is applied to one electrode while the other electrode is held at rf ground with a possible dc offset. (b) Three-layer linear trap made of gold-coated electrodes on alumina substrates. The rf is connected to the middle layer which is 125 μm thick, while static voltages are connected to the electrodes on the segmented outer layer that are 250 μm thick. In this experiment ions are coupled to motion in the direction z_T defined to be the x-axis in the asymmetric quadrupole shown in (a) and the z-axis in the linear trap shown in (b). 26

Figure 2.2 Vacuum apparatus enclosing the three-layer linear trap. Located inside the 4 inch diameter hemisphere chamber in the lower right corner, the trap is visible from large quartz window. The RF resonator is connected to the trap electrodes by the feedthroughs on top of the chamber. The vertical structure in the middle is the Titanium sublimation pump with the ion pump to its left. 27

Figure 2.3 A side view of the vacuum chamber. Two windows at 45^o allow laser access to the ions inside the trap. 29

Figure 2.4 Internal energy level of a $^{111}Cd^+$ ion. Ground state hyperfine levels (nuclear spin I = ½) serve as a qubit, with a frequency splitting of 14.5GHz. The excited states $P_{1/2}$ and $P_{3/2}$ are separated by a fine structure splitting of 72THz, and the transition from $S_{1/2}$ is resonant with a 214.5nm and a 226nm ultraviolet radiation respectively. The qubit levels |F = 0, m_F = 0> and |F = 1, m_F = 0> states are magnetic field insensitive to first order, resulting in a coherence time on the order of few seconds**.** 31

Figure 2.5 Detecting the state of two qubits with an intensified CCD camera. The images corresponds to the states shown in the figure. With CCD camera states can be differentiated which cannot be done using PMT (photo multiplier tube). 35

Figure 2.6 Quantum Logic Array architecture, (a) QCCD model in which ions are ballistically shuttled between memory region and interaction regions. (b) High level quantum computer structure where the letters Q represents the qubit and letters R

represents switch islands for quantum data communications. (c) The building block of the QLA architecture 4 level 1 building blocks are shown where the far right side outlines a single block. The solid circles are data ions and clear circles are cooling ions. 37

Figure 2.7 Bloch Sphere. (a) A qubit $|\psi\rangle$ is represented as a vector **r** from the origin to the point on the sphere with a radius of one. θ is the angle of the vector **r** with the z axis and φ is the angle of the projection of vector in xy plane with the x axis. (b) A qubit in the superposition state. (c) A qubit can be in one of the basis states 0 or 1, or in a superposition state. 41

Figure 2.9 (a) Controlled Square Root of NOT (or V gate), (b) Phase Gate, (c) pseudo-Hadamard and (d) inverse pseudo-Hadamard. 43

Figure 2.10 Controlled gates. (a) Controlled Hadamard gate, (b) Controlled Rotation with respect to angle θ. This symbol applies to any angle, particularly X, Y and Z. Additional symbol is used to denote the angle, (c) symbol of Pauli rotation where subscript i = X,Y,Z, (d) controlled phase and its unitary matrix. 44

Figure 2.11 (a) Controlled Z and its unitary matrix, (b) CNOT realized with controlled Z and Hadamard gates, (c) CNOT realized with controlled-Z and pseudo-hadamard gates. Symbol h stands for pseudo-hadamard gate and symbol h^{-1} for inverse pseudo-hadamard gate. 44

Figure 2.12 Realization on controlled phase gate. 45

Figure 2.13 (a), Controlled V gate and its equivalent realization with controlled phase gate surrounded by Hadamard gates, (b) Controlled V^+ gate and its equivalent realization with inverse of phase gate surrounded by Hadamard gates, (c) Controlled V^+ realization with inverse of phase gate surrounded by inverse pseudo-hadamard gate and pseudo-hadamard gate. 45

Figure 2.14 Example how to calculate unitary matrices of generalized rotations from general matrix formulas in Table 2.1. 47

Figure 2.15 (a) Equivalent transformation of Z gate, (b) equivalent transformation of CNOT and Hadamard gates. 48

Figure 2.16 (a) CNOT and NOT transformation, (b) CNOTs and Pauli Y transformation.

48

Figure 2.17 (a) Controlled-Z gate surrounded by pseudo-hadamards, (b) Calculation of unitary matrix for the target qubit of this gate. 48

Figure 2.18 (a) Various gates realized by ϕ for angles 0 ^{o}, 90 ^{o}, -90 ^{o} and 180^{o} in X rotations. (b) gates realized by Y rotations. 48

Figure 2.19 Explanation of operation of the Square-Root-of-NOT gate. 49

Figure 2.20 Operation of gates V and V^{+} on binary states $|0\rangle$ *and* $|1\rangle$ and quantum states $|V_0\rangle$ *and* $|V_1\rangle$. 49

Figure 2.21 Representation of one qubit, two qubit, three qubit and four qubit quantum binary gates and their classical counterpart gates. Observe the use of undesirable ancilla qubits in most gates. 50

Figure 2.22 1-qutrit ternary permutative transforms and symbolic representation. 54

Figure 2.22 Unitary matrix for Muthukrishnan-Stroud gate. 59

Figure 2.23. Symbol of ternary Muthukrishnan-Stroud gate. 59

Figure 2.24 Ternary 2-qudit Feynman gate. (a) Ternary map for qutrit Y_2, (b) realization of Y_2 using the quantum multiplexer formalism, (c) Circuit with ternary M-S gates that corresponds to the quantum multiplexer circuit from Figure 2.24(b), (d) the symbol schematic of the ternary Feynman gate. Modulo-3 addition is used. The symbol is the same as in binary logic so the reader has to observe the context. 60

Figure 2.25 3-qutrit ternary Toffoli gate (a) the general schematic, (b) its realization using ternary M-S gates (c) the ternary map of double-controlled (02) operation, (d) the schematic of double-controlled (02) gate – this is like one of Toffoli-like gate in ternary logic. 62

Figure 2.26 4-qutrit ternary Toffoli gate. (a) symbolic schematic, (b) its realization with M-S ternary gates and two ancilla qutrits. Please analyze the role of the mirror circuit at the right that restores constants 0 in two qutrits, allowing them to be reused in the next gates of this cascade. 63

Figure 2.27 A 3-qutrit generalized ternary Toffoli gate with arbitrary controlling values Operator U is executed only if the first qudit X_1 has value x1 and the second qudit X_2 has value x2. (a) general schematic, (b) its realization with the schematic of the ternary Toffoli controlled with values 2 from Figure 2.25(a). 64

Figure 2.28 (a) Fredkin gate (controlled SWAP) (b) SWAP gate (c) Double SWAP gate (d) double controlled SWAP gate. 65

Figure 2.29 Realization of a single minterm of three ternary variables. Letters x_1, x_2 and x_3 denote the controlling values of variables X_1, X_2 and X_3, respectively. 66

Figure 2.30 Realization of carry output C of ternary half-adder. (a) the ternary map of C, (b) its realization with double-controlled ternary Toffoli gates. 67

Figure 2.31 Realization of mixed binary-ternary gate using (a) symbolic double-controlled gate with binary qubit X_1 and ternary qutrit X_2. 71

Figure 2.32 Mixed (hybrid) quantum muxes. (a) binary control and binary data and its realization from standard binary controlled quantum gates, (b) ternary control and ternary data and its realization from standard M-S gates controlled with values 0, 1, 2 respectively, (c) binary control and ternary data and its realization with M-S gates , (d) ternary control and binary data and its realization with M-S gates. 72

Figure 3.1 Algebras and logic families that are of interest to this dissertation. At this stage the research on Min-Modsum and Rings is less developed but we believe that similar properties will be found in other families. 85

Figure 3.2 (a) Expansion tree with Positive Davio Expansion only, (b) An example of diagram with positive Davio gates (c) A quantum array directly corresponding to the classical diagram from Figure 3.2 (b). All Toffoli gates are 3x3 Toffoli gates only. One ancilla bit is used. 87

Figure 3.3 (a) AND-XOR circuit implementation of PPRM function in Example 3.3.1 (b) Positive Davio Decision Diagram presentation of the function (c) Quantum Array realization of the expressions. 88

Figure 3.4 (a) Fixed polarity Davio Decision Diagram presentation of the function (b) AND-XOR circuit implementation of FPRM function in Example 3.2 (c) Quantum Array oracle realization of the FPRM obtained from the Decision Diagram from Figure 3.4a. As this is a quantum oracle, the input variable values are restored. 89

Figure 3.5 Example of a KRO Expansion tree. 91

Figure 3.6 Example of a PSDRM tree. 92

Figure 3.7 $f(x_1, x_2, x_3, x_4) = x_1\, x_2\, x_3\, x_4 \oplus \overline{x_1}\, \overline{x_2}\, \overline{x_3}\, \overline{x_4}$ in certain (one of many) PSDRM form. The flattened form $f(x_1, x_2, x_3, x_4) = x_1\, x_2\, x_3\, x_4 \oplus \overline{x_1}\, \overline{x_2}\, \overline{x_3}\, \overline{x_4}$ is illustrated in Figure 3.8. 93

Figure 3.8 AND-XOR circuit implementation of function from Example 3.5. 93

Figure 3.9 A quantum array being not an oracle for flattened PSDRM from Figure 3.7. 94

Figure 4.1 This example illustrates the nature of a problem with the linear Ion Trap. A 4×4 Toffoli gate that looks like a cheap gate is, however, quite expensive when mapped to linear-neighborhood quantum array. (a) symbol of a gate as used by other authors, (b) its decomposition to Toffoli gates using one ancilla bit, (c) the final circuit with 2-qubit quantum primitives, but not-realizable in linear neighborhood as it has wires going over gates, (d) steps to realize the gate with a wire going over it, (e) the final circuit in linear neighborhood Ion Trap for the initial gate from Figure 4.1a. The number of gates in this circuit reflects the true quantum cost of the gate (macro) from Figure 4.1a. 125

Figure 4.2 Example of Lattice diagram realization in quantum. (a) Schematic of a Lattice diagram using standard multiplexers, (b) First attempt to map the schematic to 4×4 grid of 2D quantum layout. Each dot represents an ion and each oval rectangle represents the (time evolving) internal qubits of the multiplexers. Although it looks like fan-out of the middle bottom mux is two, it really is one thanks to go-through signals in some gates. Ancilla bits are not shown. This method can also be applied to the new types of reversible lattices introduced in next chapters. 126

Figure 4.3 Lattice diagram realization in 2D Ion Trap. (a) The abstract grid with cells, (b) Its placement assuming individual (serial or parallel) access (control) of all input variables corresponding to individual qubits. Sufficiently high decoherence (typical for an Ion Trap) is assumed. Mapping of input variables is not shown in (a) or in (b). 128

Figure 4.4 Initial lattice layout for 2D Ion Trap technology with width 2 of channels for buses. 129

Figure 4.5 Improved layout for a lattice with width 1 of channels. Each dot is an ion. We have to analyze if some future improvements can be done to this layout. My goal is to improve this circuit, making it more compact by some small overlap of connections and logic in cells. 131

Figure 4.6 The realization of mux on 4 qubits with 4×4 grid. (a) classical multiplexer realized in quantum array with one ancilla qubit, (b) realization of the first Toffoli from Figure 4.6a, (c) the neighborhood for the ions realizing the mux from Figure 4.6a, (d) mapping of Figure 4.6b with respect to the neighborhood pattern from Figure 4.6c, (e) mapping of the second Toffoli gate from Figure 4.6a to neighborhood pattern from Figure 4.6c. 133

Figure 4.7 Reversible Net structure to generate all multi-output symmetric functions of variables a, b, c. S1 is the net of symmetric indices from which all symmetric functions can be created. Block C^{-1} is the inverse (mirror) of block C. 134

Figure 4.8 Standard quantum array (with dimension of time from left to right) for part of the reversible Net Figure 4.7. Observe here the unfortunate lengthening of connections between qubits. The members below the EXOR gates show the maximum distance in all Toffoli and Feynman gates. The block C is shown, the mirror circuit with C^{-1} is not shown. 134

Figure 4.9 Realization of regular structures called Nets in 2D quantum layout. (a) The general structure with regular and short connections, (b) Its mapping to 2D 4×4 grid or 8×8 grid of ions. There is no need for buses as inputs are initialized only once on the top. 135

Figure 4.10 Lattice of all permutations of elements {1, 2, 3, 4}. It is used as a space of all orders of expansion variables for decision trees, decision diagrams and Net structures in optimization algorithms based on Nature. (a) the lattice in which strings in levels correspond to single transpositions (swaps) of their parent strings, (b) illustration of symmetry and numbers of elements in levels of the lattice, (c) the distance of the levels of the lattice form the initial permutation string 1234. 135

Figure 5.1.1 Shannon Tree 139

Figure 5.1.2 Kronecker Tree 139

Figure 5.1.3 Pseudo-Kronecker Tree 139

Figure 5.1.4 Positive Davio Tree 139

Figure 5.1.5 Examples of regular layouts. (a) 1D array, called also 1D layout, (b) 2×n 2D layout with 4 neighbors for very node, (c) Mesh 2D layout, (d) hexagonal layout with three neighbors for each node, (e) hexagonal layout with 6 neighbors for every node, (f) layout used in quantum dot cellular automata. Observe that on every non-oriented edge we can run information in one or two directions, so every edge represents one arrow (with any direction) or two arrows with opposite directions. The number of neighbors is thus a different parameter than grid connectivity, explained below. 141

Figure 5.2.1 A Lattice (symbols A in nodes) combined with a crippled tree (symbol B in nodes). The bottom left node represents the function output. This figure demonstrates how two regular diagrams of various types can be combined in a mapping to realize a function in a 2×2 2D regular layout. 142

Figure 5.2.2 Another way of drawing a lattice diagram. This has to be compared for 2D Ion Trap layout with the design shown at the end of the Chapter 4. Observe the rectangular grid of dots. One half of the dots (gray ones) are used for communication only. This should be reduced. Every node has 3 predecessors and 3 successors; we thus say that a 3×3 grid is used. The neighborhood number in this example is 6. Two grid dimensions are used for the input variable bus. 143

Figure 5.2.3 Binary tree (a) standard notation, (b) standard H-tree wastes spaces shown as grey circles. Nodes are enumerated for comparison. 146

Figure 5.2.4 H-tree in logic design. (a) binary tree in standard notation, (b) H-tree, wasted space is drawn in shaded circles. Numbers corresponds to nodes. 147

Figure 5.2.5 Illustration of constraints in drawing binary trees based on small H-trees. Numbers correspond to levels of the tree. Starting from level 5 of the mapped diagram we have no space in the regular layout 2x2 graph to insert all exponentially growing number of nodes. This is shown with node A that has two successors and is this not a tree. But we can insert these nodes if the tree is incomplete, which often takes place. 148

Figure 5.3.1 Regular realization of trees. An (incomplete) tree drawn on a 4×4 grid. Observe nodes a, b and c that have no space for both the predecessor nodes. The connections buses for levels 6 and 7 are also drawn. This structure shows that assuming some underlying regular substrate, certain sparse trees can be mapped to it, depending on their size. Several small trees with another neighborhood constraint (4×4 grid) are shown in Figure 5.3.1c. We call the trees that can be mapped the “crippled trees”. 149

Figure 5.3.2 Crippled tree growing only to the North and East. No arrows starting from x can be completed. A 2×2 grid is used. 150

Figure 5.3.3 Crippled tree growing North, East and West. No arrows starting from x can be completed. 150

Figure 5.3.5 (a) Mixed 1×2/2×1 grid, (b) 2×2 grid, (c) 3×3 grid different than in Figure 5.3.4b. 151

Figure 5.3.4 Basic grids, (a) 2×2, (b) 3×3, (c) 4×4, (d) 8×8. 151

Figure 5.3.6 Various regular structures:(a) Generalized PLA of the first type, (b)Generalized PLA of the second type, (c) 2×2 pattern without input buses, (d) 3×3 pattern of FPAA and a computer network of type "mesh", note the direction of arrows, (e) 3×3 pattern of a regular diagram for binary logic with oblique buses, (f) 3×3 pattern of regular diagram with two horizontal and two vertical buses (Fine Grain FPGA architecture by company Concurrent Logic Inc.), 7×7 grid, (g) 6×6 pattern of regular diagram with horizontal, vertical and oblique buses, (h) three level PLA for TANT. 152

Figure 5.4.1 Forward and Reverse Expansions in 3×3 and 4×4 structures (controlling input variables in buses are not shown): (a) Shannon in binary logic, (b) Ternary Shannon-like expansion in; a generalization of binary Shannon. 156

Figure 5.4.2 A Quantum Array that corresponds to forward and reverse expansions from Figure 5.4.1a is used. The only gates used are inverters and 3 × 3 Toffoli gates. Thus large k×k (k>3) Toffoli gates are avoided! 156

Figure 5.4.3 Symbolic schematics of quantum array structure based on 2D Ion Trap for the diagram from Figure 5.4.1a. In contrast to standard quantum array where horizontal axis corresponds to time, here both axes correspond to space and time is not shown. 157

Figure 5.4.4 Notations for 2D Ion Trap realization of upper row of the diagram from Figure 5.4.1a. (a) Dots corresponds to ions, (b) vertical axis corresponds to time inside every expansion node dot. Observe black dots at the bottom representing the next level. 157

Figure 5.4.5. Ternary quantum array to realize the diagram from Figure 5.4.1b. Observe the realization of Post literals in left lower corner. 158

Figure 5.4.6 Explanation of trees inside regular layouts. (a) the circuit obtained directly from mapping, $r = r \oplus u \oplus at \oplus at \oplus at \oplus sa \oplus sa \oplus sa$, $r = r \oplus at \oplus au \oplus sa$, (b) the same circuit after simplifications, $r = r \oplus a(s \oplus t \oplus u)$. 158

Figure 5.4.7 Quantum array for diagram from Figure 5.4.6a. The only gates used are the 3×3 Toffoli gates. 158

Figure 5.4.8 Binary quantum array for the simplified diagram from Figure 5.4.6b. The only gates used are the 3×3 Toffoli gates and 2×2 Feynman gates. 158

Figure 5.4.9 (a) The mux symbol in Akers Array notation, (b) standard mux symbol, (c) standard mux controlled by an inverted control variable $\bar{a}$, (d) another notation for inverted control, (e) one more notation for Shannon expansion, (f) circuit realization of Figure 5.4.9e. 159

Figure 5.4.10 Davio expansions in classical notation (a) Negative Davio, (b) Positive Davio, variable a is a control variable. 159

Figure 5.4.11 Expansions for Shannon, Positive Davio, Negative Davio of binary logic used in Lattices. 159

Figure 5.4.12 Comparison of addition operators in ternary algebras. 160

Figure 5.5.1 Comparison of three types of Regular Diagrams for a two-output EXOR/XNOR function. (a) Shannon lattice with no internal inverters and 3×3 grid, (b) Shannon lattice with rotated nodes and 4×4 grid, (c) Positive Davio lattice and 3×3 grid (input variable buses are not shown). 161

Figure 5.5.2 Creation of a Positive Davio level in a Regular Diagram: (a) two expanded nodes before reverse expansion, (b) layer of regular diagram after reverse expansion of nodes g_2 and h_0, (c) Fixed-Polarity RM Regular Diagram for functions f, g, h. Input buses are not shown for simplification. 163

Figure 6.1 (a) Representation of EXOR gate as a reversible gate, Feynman Gate, (b) Representation of Positive Davio gate as a 3×3 Toffoli gate. 168

Figure 6.2 The enumeration of cells of the Akers array. 169

Figure 6.3 Positive Davio lattice for an example function. 172

Figure 6.4 Algorithm for implementation of the Lattice Diagrams. 174

Figure 6.5 Pseudo-code for KFDD to QA algorithm. 176

Figure 6.6 Quantum array for the Positive Davio Lattice from Figure 3. 177

Figure 6.7 Representation of Positive Davio Lattice. 182

Figure 6.8 (a) Six variable symmetric function (b) KFDD created for six variable symmetric function (c) quantum array created for the six variable symmetric function. 183

Figure 6.9 Flipped Positive Davio Node. 185

Figure 6.10 Representation of six symmetries. 185

Figure 6.11 Example of a window permutation algorithm. 186

Figure 6.12 Example of Sifting algorithm. 187

Figure 7.1 Shannon Expansions and Reverse Expansions for Incomplete multi-output binary function. 189

Figure 7.2 Lattice Diagrams in Quantum Array, (a) the standard Shannon Lattice, (b) its realization in quantum array without mirrors. Groups of gates corresponding to (non-scalable) multiplexers are shown. This figure demonstrates that fan-out can be realized without additional ancilla bits in some regular cases. 191

Figure 7.3 (a) d-gate using traditional components, (b) representation of d-gate with Toffoli and Feynman gate, (c) d-gate with negated variable, (d) unitary matrix for d-gate from Figure 7.3b, (e)truth table for d-gate from Figure 7.3b. 193

Figure 7.4 Calculation of inverse gate to d-gate. 194

Figure 7.5 (a) Various d-gates (also known as d-gate family), (b) Representation generalized d-gate with one control line. 195

Figure 7.6 Transformation of function from classical Positive Davio Lattice to a quantum array with Toffoli and SWAP gates. Each SWAP gate is next replaced with 3 Feynman gate, (a) intermediate form, (b) final Quantum Array. 197

Figure 7.7 A view of regular layout with Dipal Gates. (a) the structure drawn similar to lattice diagrams, (b) the corresponding quantum array with SWAP gates added, (c) the general pattern abstracted from Figure 7.7b that uses d-gates and Swaps being parts of d-gate Family. 198

Figure 7.8 Another view of regular layout with Fredkin gates. (a) the structure drown similar to lattice diagrams, (b) the corresponding quantum array with SWAP gates added, (c) the general pattern abstracted from a lattice diagram similar to that from Fig. 7b that uses Fredkin Gates and SWAP gates being parts of a Fredkin Gates. 199

Figure 7.9 The general pattern of layered diagram using d-gate Family. 190

Figure 7.9 Algorithm for implementation of the Shannon Lattice Diagrams. 191

Figure 8.1 Examples of regular diagrams: (a) standard 2×2 array with only local connections, (b) standard 3×3 array with only local connections, (c) 2×2 array with two vertical and two horizontal buses per cell and the programming of its buses. Thus the grid is 6×6. (d) Akers Array with buses, 3×3 grid. These buses may be fat increasing the grid pattern size. (e) regular array with 4×4 grid and no buses. (f) a regular array with 3×3 grid and individual single-variable control of each cell. Irregular routing of these signals is not included to grid size calculation here. 193

Figure 8.2 Comparison of binary and ternary gates used in expansions. 193

Figure 8.3 Generalizations of expansions, trees, and regular diagrams from binary to ternary. 194

Figure 8.4 Mappings of various regular diagrams to one-, two- and three-dimensional regular layouts. 195

Figure 9.1 Hierarchical Decomposition and Synthesis of Ternary gates. This figure shows the plan of our research in this chapter. 197

Figure 9.1.1 (a) Classical logic diagram of GRM Form for function from Example 9.1.1, (b) Quantum array for the same form. 203

Figure 9.1.2 All AND-EXOR Forms (non-canonical forms indicated with a dashed line). 204

Figure 9.3.1 Addition (EXOR) and multiplication (AND) tables for GF(2). 204

Figure 9.3.2 Addition and multiplication tables for GF(3). 204

Figure 9.3.3 (a) Table for Addition (EXOR-like group operations) in GF(4), (b) Table for Multiplication in GF(4). 220

Figure 9.3.6 Inversions in GF(3). 208

Figure 9.3.7 Explanation of the circular property of shift gates in ternary logic. 208

Figure 9.3.8 GF(3) variable x shown with shift (inversion) operation. 209

Figure 9.3.9 GF(3) variable x shown with and without exponent. 210

Figure 9.3.10 Post Literals of height 1 (left) and 2 (right). The first will be also called the reduced Post Literal. 212

Figure 9.3.11 212

Figure 9.3.12 212

Figure 9.3.13 213

Figure 9.3.14 Shannon Tree for binary logic of two variables. 213

Figure 9.3.15 GF-Shannon Post Logic Tree for ternary logic. 217

Figure 9.3.16 Examples of some polynomials in GF(3) and their meaning. 218

Figure 9.3.17 Realization of all useful reversible ternary functions of a single variable. This figure introduces also the notations. 219

Figure 9.3.18 Realization of Post literals and Reduced Post literals (a) Post literal ^{0}x, (b) reduced Post literal. As we see, ancilla bit is necessary in both cases. 220

Figure 9.3.19 GF(3) Davio-0 Expansion Tree. 220

Figure 9.3.20 Ternary quantum array that realizes the GF(3)Davio-0 expansion from. 221

Figure 9.3.21 GF(3) Davio-1 Expansion Tree. 221

Figure 9.3.22 Ternary quantum array that realizes the GF(3) Davio-1 expansions from. 222

Figure 9.3.23 Expansion tree for GF(3) Davio-2. 223

Figure 9.3.24 Ternary quantum array that realizes the GF(3) Davio-2 expansion from 223

Figure 9.3.25 Comparison of ternary GF Davio expansions. 223

Figure 9.3.26 GF(3)-Pseudo-Kronecker Expansions. 225

Figure 9.3.27 A map of a ternary function of three variables. 228

Figure 9.3.28 Complete ternary tree that uses Galois Field Davio Expansions. It expands the ternary function from Figure 9.3.27. 229

Figure 9.3.29 Another ternary tree. 235

Figure 9.3.31 Explanation of the placement of a ternary Shannon Expansion in three dimensional space. Of course, this types of expansions and their 3D layouts can be done for any type of Shannon expansions for logic of radix 3. 237

Figure 9.4.1 GF-KRO for GF(3). 238

Figure 9.4.2 GF-PKRO for GF(3). 239

Figure 9.4.3 GF-Free for GF(3). 240

Figure 9.5.1 The Galois Field Logic Hierarchy. 241

Figure 9.6.1 Realization of Ternary multiplication for Galois Field (3) logic. This is modulo multiplication for prime number. 242

Figure 9.6.4 Synthesis of Ternary Fredkin gate by using analogy to binary case and Galois Field (3) algebraic transformations. (a) Fredkin-like ternary gate proposal controlled by value of Generalized Post Literal $^{1,2}|a\rangle$, (b) Fredkin-like proposal ternary gate proposal controlled by Post literal $^{0}|a\rangle$, (c) Fredkin-like gate proposal with MS type controls, (d) calculation of swaps for various values of controls, (e) another type of Fredkin generalization to ternary. These types of gates switch between various affine functions of two variables and generalize Dipal Gate family from binary to ternary. Symbol + is modulo addition. 243

Figure 9.8.1 Example of Generalized PLAs: Orthogonal Expansions in regular PLA-like structures: (a) General structure, (b) orthogonal column with addition and multiplication

operators, (c) orthogonal column with multiplication (Galois multiplication, AND, etc) operators. 246

Figure 9.9.1 The left side of the Reverse Expansion for Davio-0 Expansion. 247

Figure 9.9.2 The right side of the Reverse Expansion for Davio-0 Expansion. 247

Figure 9.9.3 First three levels of a ternary lattice. The lowest level is for variable c. 248

Figure 9.9.4 Regular 3-dimensional lattice for ternary expansions. Every circle shows location of a group of ions in 3D space and letters correspond to ordering of input variables. This diagram illustrates 3-dimensional symmetries of ternary functions. 248

Figure 9.9.5 A small tree of arbitrary non-reversible operators to be mapped. 249

Figure 9.9.6 Mapping to layout with mirrors (a) arbitrary fat tree, (b) its mapping with mirror operators. 249

CHAPTER 1

Introduction.

There exists a growing body of papers about Computer Aided Design of reversible and quantum circuits. Most of these papers are related to designing and optimizing circuits in binary quantum computing. The area of logic synthesis for multiple-valued quantum circuits was started by my advisor professor Marek Perkowski in the year 2000, and the first paper was published in 2001. When I was charged with the task of developing algorithms and software for synthesis of reversible quantum circuits, there was no literature available. This situation changed in the years 2003-2009 when several papers were published by Marek Perkowski, Xiaoyu Song, Anas Al-Rabadi, Lee Casperson, Erik Curtis, Michael Miller, Dr. Bullock, Faisal Khan, and Mozammel Khan. There were also many papers published on binary quantum circuit synthesis and general-purpose multiple-valued logic that I took into account when working on my subject. All synthesis problems related to reversible logic are NP hard and the solutions are so far restricted to rather small circuits. Because these problems are difficult, the algorithms to be developed should be approximate, rather than exact. (NP hard are optimization problems that are counterparts of NP complete problems. NP complete problems as far as known require exponential complexity to find the solution but polynomial complexity to verify it.)

Let me explain the motivation for this dissertation, called "Design of Regular Reversible Quantum Circuits". During the development of my M.S. thesis I investigated regular structures for FPGAs (Field Programmable Gate Arrays) and testing of such structures in VLSI (Very Large Scale of Integration). It is therefore a natural extension of my previous

work that the regular structures, known to be very promising in many forthcoming nanotechnologies and in quantum circuits in particular, will be the subject of my research. After developing methods for binary quantum circuits I became interested in multiple-valued quantum circuits, because this is the first available technology where building multiple-valued circuits is of similar complexity as building binary circuits – it was not so in the previous realization technologies such as CMOS. In the past, a large body of theory on multiple-valued logic, synthesis methods and circuits was accumulated (ISMVL conferences and "MVL and Soft Computing" Journal) with practically little use to design circuits. The field of synthesis of multiple-valued reversible circuits started with papers [DeVos02, Al-Rabadi01, Perkowski02] and the field of study becomes even more difficult when one attempts to synthesize and minimize multiple-valued (MV) Quantum circuits [Muthukrishnan00, Perkowski05]. Most of the current algorithms are predominantly based on heuristic and evolutionary ideas, while a few algorithms are based on matrix algebra or group theory. All these algorithms are applicable only to very small circuit specifications, and most of the current research efforts assume existence of no ancilla bits. I think that this is not necessarily a good assumption since a few ancilla bits do not cost much even with the current high costs of quantum bits (the width of the "quantum register"). These few qubits will have even less meaning when quantum technology advances. One more aspect of my interest was design for test of quantum circuits. It is well known how important testing of classical digital circuits is and as a Design for Test (DFT) engineer for Intel and previously at Motorola and Mentor Graphics I have extensive experience with testing and test theories for the currently

available technologies. I was thus interested in developing methods that would produce highly testable circuits.

Currently there are not many papers on synthesizing and optimizing circuits for Ion Trap Quantum circuits and computers, the recently invented breakthrough technology that is praised as a leading future technology by top specialists. In this dissertation a new approach for solving several hard problems in Computer Aided Design (CAD) for such computers, and particularly logic synthesis of quantum circuits, and even more particularly permutative circuits and oracles realized in Ion Tap computers, is given. Oracles are the most important part of the famous algorithm by Grover and other similar algorithms. They answer only yes/no questions. The applications of Grover's algorithm to solve various NP-hard problems changes by designing new oracles for them, so the question is how to design these oracles at all, and more importantly how to design them efficiently. The permutative circuits that are not oracles have the same number of inputs and outputs, and the inputs are not necessarily repeated on outputs, as it is required in oracles. They are useful to build parts of oracles. For instance the spectral transforms, arithmetic blocks, image processing blocks, calculations of cost functions, "counting number of ones in input vector" and many other similar tasks (that are normally built into the data path of a standard or signal processing computer) all belong to this group. We also deal with fast quantum transforms, another important class of algorithms by itself, used for instance in the famous Shor's algorithm or in those quantum computers for vision which use the Fast Quantum Fourier Transform. Thus the thesis covers design of two kinds of blocks so far used in quantum computing: binary and multiple-valued.

Mixed (hybrid) and analog (continuous, fuzzy) quantum circuits (which seem to be too far from the methodologies developed herein) are not discussed in this dissertation.

Based on works on Ion Trap Quantum Computing [Wineland98, Garcia-Ripoll03, PHDLee], quantum teleportation [Steane97, Milburn00] and ideas of the architecture of quantum computers [Metodi05] proposed by Chuang (the creator of the quantum computer with most number of qubits until year 2008), I develop a model for circuits in a hypothetical, yet to be built multiple-valued quantum computer in Ion Trap technology. Although I borrow some ideas from the literature, my design is original. Whenever I was influenced by some previous work, the text of the dissertation clearly describes what is new and what is taken from other sources. Chapters 2, 3 and 4 are mostly based on the literature. The remaining chapters present my original work.

Next we analyze possible application areas of such computers and their advantages over a binary NMR (Nuclear Magnetic Resonance) quantum computer as discussed in [PHDSazzad]. There is of course much speculation in my dissertation related to a non-existent technology as of 2010, since only NMR computers for 10 bits exist and there are no Ion Trap Quantum Computers with that many qubits yet. Recent information from the press and scientific journals about Ion Trap computers is therefore used to make accurate predictions based on the most modern state of the art developed in year 2010. When it comes to applications of these new computers, we are particularly interested in their uses in solving NP-complete and NP-hard discrete problems, computer vision and robotics. It has to be pointed out that although we speculate on the existence of a quantum computer

with hundreds or even thousands of qubits (and qudits for MV quantum computers), the physical reality of all quantum phenomena such as quantum parallelism, superposition and entanglement required to build quantum computers is no longer a matter of speculation. This is because they have already been experimentally verified. For instance the Grover and Shor algorithms have already been experimentally verified. The interested reader is referred to the growing literature on Ion Trap computing and especially to an excellent introduction in [Milburn00, Steane97, Leibfried03, Monroe95].

Multiple-valued logic is not used much in any hardware designed with classical logic and only very few quantum algorithms that use MV signals are known at this time. In this dissertation I will develop the research on designing such circuits, to be implemented in a practical quantum technology. Many generic algorithms used in classical computing such as satisfiability, graph coloring, scheduling and others can be either generalized to multiple-valued logic or naturally formulated in it. This dissertation creates methods that can be used to implement practical quantum algorithms realized on future Ion Trap quantum Computers. These algorithms include: quantum covering (a generalization of the even-odd covering problem), unate covering (classical), graph coloring, scheduling, vision transforms, pattern classification problems and several others.

Ion Trap quantum computers will find various practical applications, including those in the area of quantum computational intelligence. Quantum computers will be used to solve a variety of combinatorial problems and will become a basis of future intelligent robotic and media systems. To aid in inventing these future algorithms a new multi-valued

quantum logic system is invented and investigated in this dissertation. I introduce Galois Field and Controlled Gate quantum logics. These new logic algebras should be of interest to the quantum logic synthesis community because of their analogies and extensions to the classical Reed-Muller Logic and classical reversible logic of Feynman, Toffoli and Fredkin [Feynman82, Fredkin82, Toffoli90, Feynman96], and hence their properties of high testability. Biamonte and Perkowski extended the classical Reed-Muller logic algorithm based synthesis method of Reddy to quantum circuits, using an equivalent of Positive Polarity Reed-Muller logic. Sarabi and Perkowski [Perkowski92a, Perkowski95], Sasao, Bhattacharya and others generalized the results of Reddy and Pradhan to more complex circuit structures than Positive Polarity Reed-Muller forms. In this thesis I will show how Kronecker Decision Diagrams and Lattices, Fixed Polarity Reed-Muller forms (FPRM), Generalized Reed-Muller forms (GRM), and Exclusive-Or-Sum-of-Products (ESOP) binary circuits can be generalized to multiple-valued quantum circuits. I will discuss classes of MV expansions, canonical forms, trees and diagrams as well as corresponding flattened expressions and circuits in their quantum realizations. In this sense this thesis is a continuation of the PhD theses of Anas Al-Rabadi and Sazzad Hossein, the M.S. thesis of Jeff Allen and the journal papers of Jake Biamonte and Prof. Perkowski. I do not analyze in detail the testability of the new circuits but because they use group logic properties, they are similar to circuits analyzed in papers of Biamonte, Perkowski and Allen.

Design implementation of multiple-valued quantum circuits with quantum arrays using multiple-valued Muthukrishnan-Stroud gates [Muthukrishnan00] in Ion Trap and similar

technologies recently became practically possible. Moreover, based on generalizations of binary circuits to multiple-valued circuit families shown in the literature [Perkowski97, Perkowski97d, Perkowski02, Perkowski05, Pierzchala94, Sasao78] and extended in this dissertation, such circuits can be practically built using new design methods proposed here. However, the algorithmic complexity of synthesizing large circuits of this type exceeds by far the complexity of designing classical circuits. Efficient heuristic methods of synthesizing them are therefore necessary. The researches of previous PhD students at PSU such as Anas Al-Rabadi [PHDAl-Rabadi] as well as researchers world-wide have been only partially successful and there exist as yet virtually no CAD tools for MV quantum computing. Isolated design programs are restricted to toy size problems. In particular, there are no logic synthesis tools for Ion Trap QC, which have certain specifics that do not exist in other circuits. In addition to the fact that the circuits discussed in this dissertation are realized in Ion Trap technology and many of them are multiple-valued, there are two other main ideas that drive my research interest: the regularity of the FPGA-like substrate of Ion Trap circuits and the related property of their high testability.

The organization of the chapters will be now outlined in more detail. I first show how new families of algebras, which are generalizations of classical binary AND-EXOR forms non-canonical expressions can be developed in multiple-valued logic. These circuit families play the role of building blocks for MV quantum circuits realized in the Ion Trap technology. The descriptions with these new logics also allow an easy conversion of non-reversible MV specifications to MV reversible specifications. A complete logical hierarchy of expansions, trees, decision diagrams, and forms for these new MV families

is developed in my dissertation, following, expanding and systematizing the methods developed in the work of Professor Marek Perkowski and his students: Richard Safranek, Konika Ganguly, Karen Dill, Edmund Pierzchala, Craig Files, Rolf Drechsler, Nouraddin Alhagi and Sazzad Hossain. These ideas are influenced on the one hand by algebraic structures, both those already used in quantum mechanics [Brassard, Brylinski] and on the other hand by the real quantum technology such as that presented in papers of Isaac Chuang, Muthukrishnan, Stroud [BrassardGalois03, Brylinski01, Muthukrishnan00].

I present a multiple-valued logic that can be realized in regular structures (examples of regular structures are classical PLAs and Akers Arrays). Regular 2D structure is a fundamental aspect of building MV quantum arrays with generalized Toffoli gates, a concept somehow similar to classical PLAs, but adapted to MV quantum circuits. Such structures can be created using Ion Trap technology where ions are located in two-dimensional or three-dimensional structures in space. In this dissertation I present ternary and quaternary logic for illustration, but some of these ideas are general and can be extended to arbitrary radices. Some logics with other radices cannot however be mapped naturally to three-dimensional spaces of our Universe. Galois Field logics exist for radices 3, 4, 5, 7, 8, 9 and so on. Because of space limitations we discuss only the cases of 3 and 4 but larger fields are similar. There exist multi-valued circuits based on Rings and other algebra, but how to map them to 3D space? This can be done only on smaller circuits, or without the perfect regularity possible for Galois Field (2) and Galois Field (3). The research for other logics becomes thus more complicated as Galois Fields do not exist for all radices and because there are only 3 geometrical dimensions in our universe.

A number of different families of algebras are presented and the presented theory is next further generalized to different types of multi-valued logics with various radices. Hybrid circuits that mix various radices have also been introduced in the literature. For some regular or quasi-regular structures such circuits can be efficiently mapped and our methods can be extended to the hybrid circuits, but these are not covered in the dissertation. Several expression types presented in the dissertation have a high degree of quantum testability. This provided me further motivation for their introduction and study. The most important multi-valued Quantum logic family is the Galois Field family (that takes its roots in mathematics). The generalized Controlled Gates family takes its roots in physics - quantum technologies and Ion Trap in particular. I will discuss the relations between the two families and how they relate to regularity of circuit structures.

Additionally, new families of logic are invented on top of these families from the insight gained by analogies of the different, presented algebras. The previous research has shown that the hierarchy of Reed-Muller Expansions can be generalized to a subset of the Linearly Independent Hierarchy of expansions [Perkowski97a, Perkowski97b, Perkowski97c]. Previous PhD students from PSU, Bogdan Falkowski, Ingo Schafer, Karen Dill, Ugur Kalay and Anas Al-Rabadi, have demonstrated how some Galois diagrams and forms [Zeng95] can be obtained by extending the Linearly Independent (LI) Reed-Muller concepts to arbitrary Galois Fields [Dill97, Dill97a, Alrabadi01]. We are able, however, to show here some further improvements and generalizations. These methods are related to modern quantum technology and some breakthroughs occurred in quantum realizations just very recently, in years 2004-2009.

Because of high algorithmic complexity, some kind of search is the basis of efficient CAD algorithms for quantum circuits. In this thesis I propose methods for ordering regular logic structures. These methods are based on trees, diagrams and forms. Search algorithms find applications for ordering gates for cascades and selecting gates for them. They are also used to find orders of expansion variables and orders of expansion types in pseudo- types of decision diagrams, trees and lattices. Search also has other applications. As in classical design, the structure, the search and the representation are the most important components of successful CAD programs for regular quantum circuits. The search algorithm technique provides a means for designing a quantum circuit using quantum gates. In case of synthesis of Reversible Logic Circuits, search parameters corresponds to various reversible gates and their input/output wires.

In this dissertation I present a new approach to creating a quantum array for designing reversible quantum circuits using Kronecker Functional Lattice Diagrams (KFDD). Any given Boolean function is first synthesized in KFDD and then converted into a quantum array. My method of creating a quantum array from the KFDD is unique and never presented in the literature before. Unlike the other contemporary algorithms for synthesis of reversible functions that always use $n \times n$ Toffoli gates, my method invariably synthesizes functions using 3×3 Toffoli gates, Feynman gates and NOT gates. My method adds a small cost by adding extra ancilla bits, however, the overall quantum cost of the circuit is reduced as the circuit is designed with smaller primitives. This method can create a quantum array for any Boolean function, whereas all other well known

methods require reversible specification of the function to synthesize a circuit. The comparison of benchmark results proved my method to be superior compared to other contemporary approaches. I also invented a family of new gates called “Dipal Gate” (name given by my advisor Dr. Marek Perkowski). This gate is a reversible counterpart of the classical multiplexer. This is generalized as a one control qubit and two target qubits reversible gate. Several variants of Dipal gate are presented in the Chapter 7. I also developed a concept of layered diagrams that efficiently use gates from the Dipal gate family. The CAD tool for creating the KFDD and creating the quantum array is developed as a part of this dissertation.

Concluding, there are several new ideas proposed in my dissertation.

1. New concepts of binary regular structures based on known and new expansions are presented.
2. The concept of creating a quantum array from a Kronecker Functional Decision Diagram (KFDD) is developed, and this method can be used for other similar diagrams. This algorithm synthesizes a reversible circuit for any Boolean function.
3. A new gate, called Dipal Gate (by Dr. Perkowski), has been invented and shown useful. Respective algorithms have been programmed and shown to be superior compared to previously known methods.
4. A multi-valued logic family of algebra is introduced. My interest is, mostly, however in multiple-valued circuits. It is presented in many examples how the concepts developed for binary quantum circuits in the first part of the

dissertation can be expanded and generalized to various multiple-valued logics.

5. Expansion methods for multiple-valued families of GFSOP circuits and general multiple-valued reversible circuits with ancilla bits have been presented. Algorithms similar to the one used for binary logic can be developed for them.

A short description of the content of the dissertation is given below.

1. Chapter 1 is an overview of the presented research. It presents the history of this thesis and its main concepts and ideas – no details just basic concepts. However, to thoroughly understand the contents of this chapter the reader should be familiar with the next chapters of the thesis first.

2. Chapter 2 is based on the literature. It presents the details of the Ion Trap technology which is one of the most promising technologies for building quantum computers. This Chapter introduces binary and multiple valued reversible gates.

3. Chapter 3 presents basic ideas of types of trees, decision diagrams, and expansions for designing quantum circuits.

4. In Chapter 4, ideas on CAD tool development for binary and multiple valued quantum circuits are presented. This chapter also introduces the drawbacks of contemporary methods for designing reversible logic circuits and provides motivation for addressing those problems by introducing new algorithms presented in Chapter 6 and Chapter 7.

5. Chapter 5 introduces concepts of regularity in binary quantum circuits.

6. Chapter 6 presents one of the main contributions of my research. In this chapter I present the idea of creating a quantum array for any Boolean function, from the Kronecker Functional Decision Diagrams (KFDD). The quantum circuits synthesized using this methods use basic Toffoli gates, Feynman gates or NOT gates. The ideas presented in this Chapter are completely new and never presented in the literature before.

7. Chapter 7 is about the new gate that I invented during the course of my research work. Dr. Perkowski called it a Dipal gate. Many variants of the Dipal gate are presented along with the concept of layered diagrams that use variants of the Dipal gate in synthesizing reversible logic circuits.

8. Chapter 8 presents basic ideas for creating valued quantum arrays.

9. Chapter 9 generalizes all my ideas from chapters 5-8 from binary to multiple valued logics. This Chapter provides insight on Galois field logic, Group logic and their use for regular quantum circuit synthesis.

10. Chapter 10 concludes the dissertation with final remarks about future work.

The order of topics, structure and flow of my dissertation is shown in Figure 1.1.

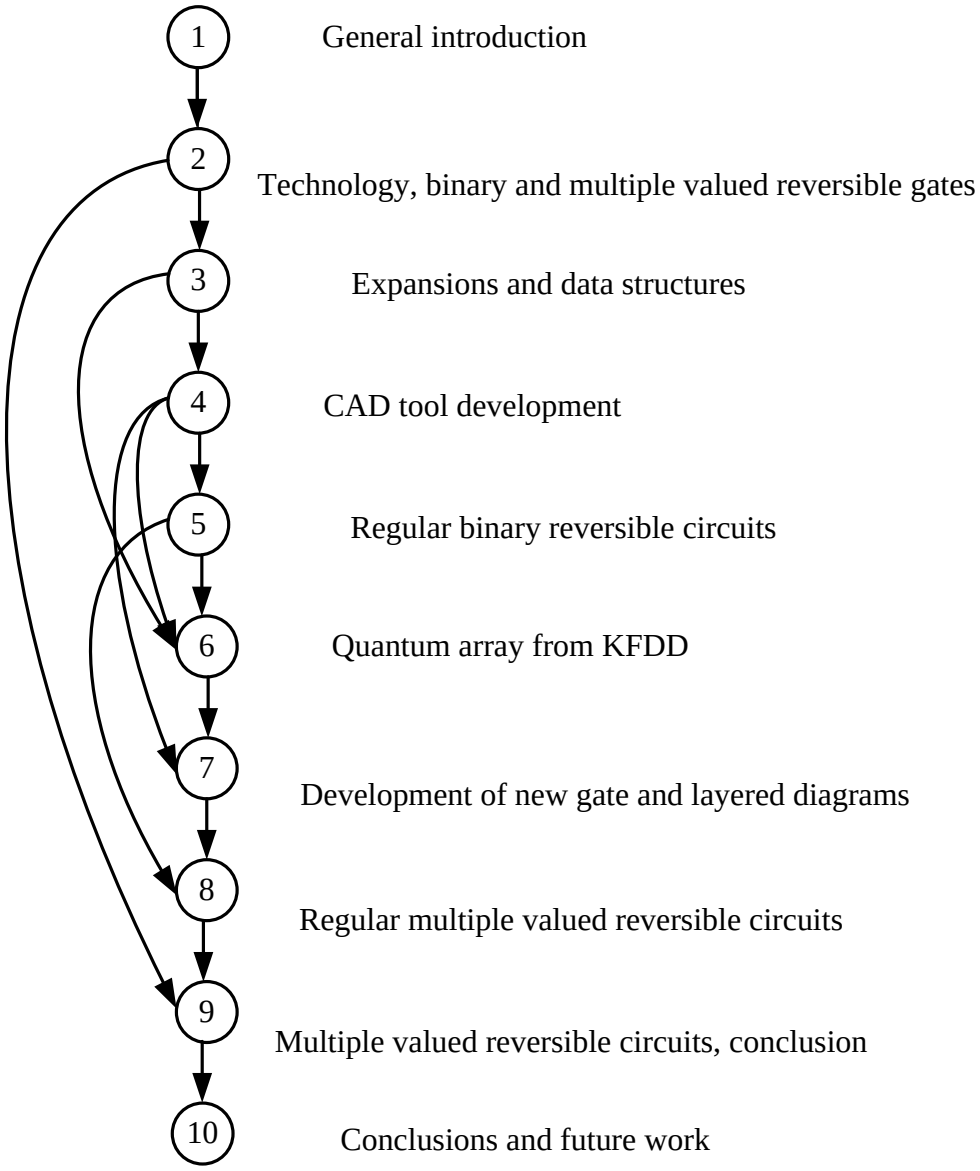


Figure 1.1. Organization of dissertation.

CHAPTER 2

Binary, Ternary and Hybrid (mixed Binary-Ternary) Quantum Gates.

2.1. Introduction.

This chapter is based on the literature and the material is taken from many sources. This chapter describes Ion Trap technology and experimental implementation of this technology. Wolfgang Paul and Hans Dehmelt, who developed the first ion trap in the 1950s, they received the Nobel Prize for their work. In 1995 Cirac and Zoller proposed a scheme to implement quantum controlled-NOT gate using trapped ions and introduced Ion Trap technology as one of the most promising technologies for quantum computers. Further in this chapter I will introduce the concepts of binary, ternary and hybrid quantum gates and quantum circuits. This is meant as an introduction to the following chapters, which will present formal descriptions of circuits and synthesis methods. The concepts of quantum mechanics and notations will be included. This chapter should motivate the reader to appreciate the concept of MV-logic design of quantum circuits for Ion Trap technology. Our treatment of binary, multiple-valued and hybrid quantum circuits will be unified**.**

2.1.1. Overview of Ion Trap technology for Quantum Computing.

The ability to trap ions in three dimensional space has contributed to many advances in atomic physics, laser cooling, and atomic clocks, as well as applications such as mass spectrometers. Quantum Computing was first introduced to ion trap system in 1995 by Cirac and Zoller. They proposed a scheme for implementing a quantum controlled-NOT gate using trapped ions by coupling the internal state of two ions in which ions interact

via their Coulomb interaction, which creates common vibrational modes of the ion string [Cirac95]. Later there were many other schemes presented for implementation of the controlled-NOT gate [Monroe95, Schnidt-Kaler03, Molmer99, Milburn00, Sackett00, Garcia-Ripoll03, Leibfried03]. The functioning of these gates relies on the concept of spin-dependent force and has many improved characteristics compared to the original Cirac and Zoller proposal. In addition, the idea of a quantum charge-coupled devices was also presented as a scalable architecture for trapped ion quantum computing, using an array of trap zones where ions can be stored and transported to specific locations for computation when required (flying ions) [Kieplinski02]. This theoretical and practical work provided a foundation for trapped ion quantum computing and a complete framework for a scalable universal quantum processor. The primary requirements for physical realization of quantum computations are given in [Cirac95]:

1. A scalable physical system with well characterized qubits
2. Ability to initialize the state of a qubit
3. Long decoherence time
4. A set of universal quantum gates
5. A qubit specific measurement capability

2.1.2. Why reversible logic circuits?

Landauer [Landauer61] proved that binary logic circuits built using traditional irreversible gates inevitably lead to energy dissipation, regardless of the technology used to realize the gates. Zhirnov *et al.* [Zhirnov03] showed that power dissipation in any future CMOS will lead to an impossible heat removal problem and thus the speeding-up

of CMOS devices will be impossible at some point which will be reached before 2020. Bennett [Bennett73] proved that for power not to be dissipated in a binary logic circuit, it is necessary that the circuit be built from the reversible gates. *A gate (or circuit) is reversible if it is a one-to-one mapping between sets of input and output values*. Thus all output vectors are just permutations of input vectors. (Such a circuit can be described by a binary permutation matrix [Nielsen00]). Bennett's theorem suggests that every future (binary) technology will have to use some kind of reversible gates in order to reduce power dissipation. This is also true for multiple-valued reversible logics, which is an additional advantage because the multi-valuedness by itself demonstrates several potential advantages over binary logic. These potentials of MV logic so far have not been taken advantage of since they bring no technological improvements when applied to existing technologies such as CMOS. All these fundamental results of Landauer, Bennett and Zhirnov are technology-independent but practically applicable to future nano-technologies, especially to quantum technology as being the most advanced of all the nano-technologies. They are also applicable in quantum dots and DNA circuit realization technologies.

2.1.3. How does reversible logic applies to quantum circuits?

Quantum technology is inherently reversible and is one of the most promising technologies for future computing systems [Nielsen00]. In addition to reversibility, it has powerful properties such as quantum superposition, quantum parallelism and quantum entanglement that allow for solving problems much more efficiently than in classical computing. For instance, while a classical algorithm needs N steps to search an

unstructured database, a quantum algorithm proposed by Grover [Grover96] for the same problem needs only $\sqrt{N}$ steps where N is the number of elements in the searched unstructured space. It can be proved, moreover that there is no classical algorithm that would require fewer steps than O(N) [Zalka99, Boyer96]. (Observe that the quantum circuit is reversible when it calculates in Hilbert Space before the measurement. It is no longer reversible after measurement, since the probabilistic measurement cannot be reversed). Although only a few quantum algorithms are known in 2010, many problems can be reduced to some of these algorithms, for instance to the Quantum Fast Fourier Transform or to Grover's algorithm. Thus, any NP-hard problem can be reduced to Grover's algorithm to give a practically useful and substantial reduction in complexity for large values of N. This reduction is, however, not as high as in the case of the exponential speedup obtained by the famous Shor's quantum algorithm [Nielsen00] for integer factorization.

2.1.4. What are oracles and why are they important in quantum computing?

An oracle is a logic circuit that answers "yes/no" to a question asked of it, for instance – "*is this mapping of nodes to colors in a non-oriented graph a correct graph coloring?*" A quantum oracle must be built from quantum gates to allow superposition and entanglement of its outputs, which is the basis of Grover's algorithm. Remember that inputs to the oracle are also repeated as some of the oracle's outputs in order to make the measurement of all inputs together with the (output) oracle qubits. The states of inputs to the quantum oracle encode the solution, using the so-called "phase kick-back" [Nielsen00]. Without going into details of how an oracle works in a quantum algorithm,

let us observe here only that the oracle must be built from quantum gates and that many oracles include special types of quantum circuits such as: arithmetic blocks, logic blocks, relational blocks, and mixed blocks [Sazzad08]. If one only know how to build a respective oracle, then with the availability of the physical quantum computers Grover's quantum algorithm (and its variants and modifications) would be immediately useful to solve very many highly important problems such as Travelling Salesman. It is therefore important to study methods and algorithms to build various types of oracles and their parts. The problem of building various classes of oracles or their blocks (components) is well-known in the case of binary quantum circuits [Nielsen00]. It was discussed in the review about automatic quantum synthesis [Perkowski04]. Many papers on how to synthesize them from binary quantum gates, or that propose general-purpose logic synthesis algorithms for binary quantum circuits, have been published [AgrawalJha04, Mishchenko02, AlRabadi04, Miller03a]. In this dissertation I will propose new methods and algorithms to build various types of quantum circuits, and specifically circuits used as oracles. Although the main results of the thesis are documented using binary logic, the ultimate goal of this research is to develop methods for ternary, hybrid and other MV quantum circuits. Some of the new MV logic synthesis methods for quantum circuits are discussed in full detail although they are not programmed. Some others are just sketched as I found that the number of new methods based on my basic principles formulated in the introduction is vey high.

2.1.5. Are multiple-level circuits realizable in quantum technologies, and if so which ones?

After year 2005 there was a growing interest in the quantum and particle Physics research communities in the realizations and applications of d-level quantum systems (a different name used by them for multiple-valued quantum logic circuits). The interest in such circuits is not only because natural quantum mechanics systems are inherently d-level and because they generalize the standard binary quantum logic, but mainly because the d-level systems improve some standard quantum circuits, algorithms and protocols.

The case of special interest in recent times is d = 3, for which the qudit is called a qutrit. This is the simplest MV logic but it is already more complicated than binary quantum logic. Some results from classical ternary logic and reversible ternary logic can be adapted. On the other hand, new important properties can be analyzed. For instance, the following facts are known when comparing the cases of d = 2 and d = 3:

1. Some cryptographic protocols such as quantum bit commitment and coin-flipping protocols have been proved to be more secure for d=3 [Peres00, Bourennane01, Spekkens02].
2. The quantum cryptographic protocols, such as those that generalize the Ekert's entanglement-based protocols for d=3, are more robust against optimal individual eavesdropping attacks, such as cloning-based attacks for d=2. They are also more robust than using un-entangled qutrits [Bruss02, Cerf02, Durt03].
3. For d=3 quantum information processing is more powerful than for $d \neq 3$ as the best use of Hilbert space dimensions is achieved [Greentree04].

4. Higher error-tolerance is achieved for fault-tolerant quantum computation for d=3 than for d=2 [Knill04].
5. Usefulness for quantum simulation [Terhal99], quantum tomography [Thew02], and quantum games [Flitney02].

There are recent publications that propose various ways of building ternary quantum circuits with existing or new quantum realization technologies. Muthukrishnan and Stroud [Muthurkrishnan00] proposed for the first time the ion trap realization technology for arbitrary radix. This paper was seminal, but it was only theoretical. Klimov et al [Klimov03] realized ion-trapped qutrit-based generalization of the quantum circuits built by Cirac and Zoller. D. McHugh and J. Twamley [McHugh05] and modified the work on ion trap quantum computers from [Mintert01, McHugh05a] to qutrits. An axial magnetic field gradient was applied across an ion chain that allows the three hyperfine Zeeman energy levels of each ion, forming the qutrit, to be individually frequency addressed. This technology demonstrated also the coupling between qutrits, thus leading to conditionally controlled gates where the state of one qutrit selects an operation executed on another qutrit. These gates are natural generalizations of those, which are fundamental to binary quantum computing such as controlled-NOT (Feynman gate). Finally, Das et al [Das03] presented experimental realization in NMR of all single-qutrit gates that I use in our designs in this dissertation. They were also able to demonstrate preparation of pseudo-pure ternary states used by such circuits. Their design was based on utilizing deuterium (spin-1) nuclei partially oriented in liquid crystalline phase.

Based on work of [Klimov03, Lawrence04, Mintert01, Nielsen02] we can conclude that there are a few existing competing technologies in which universal sets of ternary quantum gates are realizable. There are also formal Lie group theory based [Brylinski01] and software-based [Bullock05, Curtis04, Denler04a, Khan05, Yen05] synthesis methods for MV quantum circuits. The research on binary quantum blocks for oracles has started recently (2005) and there are even fewer works on multiple-valued quantum blocks – I want to fill this void in my dissertation.

2.1.6. Practical examples of multiple-valued quantum circuits.

Reversible adders from conservative gates have been presented in [Bruce02]. Quantum realization of a ternary full-adder was given for the first time by Miller *et al.* in [Miller04, Miller04a] and a quantum realization of a ternary parallel adder/subtractor with look-ahead carry was reported by Khan and Perkowski [Khan05a]. This dissertation presents exact and approximate synthesis of quantum realization of the ternary circuits, using the macro-level ternary Feynman and Toffoli gates that are built on top of the Ion Trap realizable 1-qudit and Muthukrishnan-Stroud gates [Muthurkrishnan00].

2.1.7. Costs of gates and circuits in multiple-valued quantum technologies.

As the realization experience of building ternary quantum circuits is limited, it is difficult to compare costs of gates realized in various quantum realization technologies. Counting the gate cost only in terms of the number of elementary gates may be misleading; because the costs of various gates, even in binary, differ a lot. It is also difficult to compare costs of gates with multiple-valued measurements based on qudits [Curtis04, Denler04a, Khan04, Khan04a, Khan05c, Perkowski02, Perkowski04, Yen05] and the gates realized

on encoding multiple-valued symbols to Hilbert space but with binary measurements (ternary case in [Miller04, Miller04a] and quaternary case in [Kim00]). One can theoretically compare the number of electromagnetic NMR pulses [Das03, Kempe02, Lee06] as the cost function, and evaluate the cost of observables, but these costs may differ a lot in various specific quantum realization technologies such as NMR or Ion Trap (it is an open problem how much can they differ). It is the long-term goal of the PSU quantum group to compare realization costs in various multiple-valued quantum technologies, but this topic is beyond the scope of this dissertation. More studies on practical realizations of preparation and measurement circuits in MV quantum technologies are also necessary. In this dissertation, I counted the numbers of elementary "primitive quantum" gates (i.e. directly quantum realizable "primitive" gates) or the numbers of elementary pulses as the measures of circuit's cost. In another variant I counted the numbers of rectangles (or ions) in 2-dimensional realizations.

Let us point out that Nature does not favor binary quantum states. Quantum circuits with ternary or quaternary states are therefore as natural as binary quantum circuits. In theory every unitary transformation in Hilbert space H_k can be used as a basic quantum gate. Similarly as in the binary case, I have selected some subset of gates that are easily realizable and at the same time allow the creation of a good mathematical base for synthesis. In the binary case the basic gates are Hadamard, Controlled-Square-Root-of-Not, Feynman and Toffoli. I asked myself the questions:

- "What are their counterparts in ternary?

- What are the counterparts of X, Y and Z rotations and Pauli rotations in particular? This thesis is an attempt to answer these questions.

As observed in [PHDAl-Rabadi], the mathematically useful in synthesis counterpart of the Hadamard gate is the Chrestenson gate. It performs the well-known Chrestenson Transform [Cerf02], being the special case of the Fourier Transform. For the ternary case we deal with Hilbert space H_3 and we should use analogies to Hilbert space H_2 being the base of all binary quantum gates.

Interestingly, quantum mechanics let you build "hybrid" circuits that mix binary and ternary logic. The quantum wires in such circuits correspond either to qubits (binary) or qutrits (ternary) logic. All single qubit operations in binary wires are binary and all single qutrit operations in ternary wires are ternary. Similarly, the target controlled operations in binary bits are binary and the controlled operations in ternary wires are ternary. The controls from binary wires are binary and the controls from the ternary wires are ternary. Such circuits can be synthesized using methods introduced in this dissertation.

2.2. Ion Trap technology for quantum computers.

This section describes details of the experimental setup for trapping ions in three dimensional space.

2.2.1. Ion trapping.

This section outlines the basic function of ion trapping including ion traps and their loading mechanism as well as the laser system for probing the ions. The physical system

presented here [PHDLee] contains the quantum mechanical degree of freedom which is employed for computing purpose, such as the qubit stored in the internal electronic structure of the ion and the external quantum mechanical motion of the ions in the trap available for quantum information transfer. Basic operations such as initialization and readout of the qubits can be performed using a specialized laser imaging system. The ion trap technology proposed [PHDLee] is based on the design invented by Wolfgang Paul that confines charged particles with an electric quadrupole field oscillating at radio frequency. Such traps are also called "Paul" or "rf" traps. Two types of traps are proposed in [PHDLee], a "ring-and-fork" asymmetric quadrupole trap and a three-layer linear trap as shown in Figure 2.1. In a ring and fork trap, the potential near the center of the trap can be modeled as an asymmetric 2-quadrupole potential when a rf potential $V_0\cos(\Omega_T t)$ is applied to the ring electrode and a static potential U_0 is applied to the end-caps. The potential at coordinate (x, y, z) is given by

$$V(x, y, z) = \kappa(U_0 + V_0 \cos(\Omega_T t)) \left(\frac{\alpha x^2 + (2-\alpha) y^2 - 2z^2}{d_0^2} \right)$$

Where $\alpha \sim 0.8$ and $\kappa \sim 0.8$ are parameters determined by the geometry of the electrodes, $d_0 = \sqrt{r_0^2 + 2z_0^2}$ is the characteristic internal dimension of the trap, with r_0 being the radius of the ring electrodes and $2z_0$ being the separation between the two end-caps.

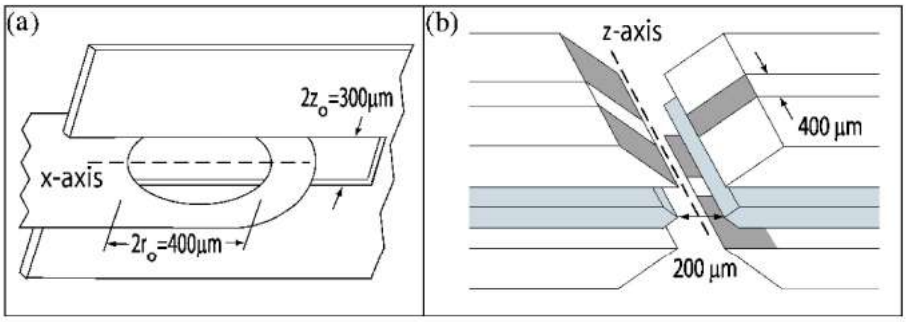


Figure 2.1 Schematic diagram of an ion trap electrode. (a) Asymmetric quadrupole trap with a 400 μm diameter ring electrode and a 300 μm gap between the form electrodes. The rf is applied to one electrode while the other electrode is held at rf ground with a possible dc offset. (b) Three-layer linear trap made of gold-coated electrodes on alumina substrates. The rf is connected to the middle layer which is 125 μm thick, while static voltages are connected to the electrodes on the segmented outer layer that are 250 μm thick. In this experiment ions are coupled to motion in the direction z_T defined to be the x-axis in the asymmetric quadrupole shown in (a) and the z-axis in the linear trap shown in (b).

The standard equation of motion with dimensionless parameter can be written as

$$\frac{d^2x}{d\tau^2} + (a + 2q\cos 2\tau)z = 0$$

Where

$$a = \frac{8eU o\alpha\kappa}{md o\,\Omega_T^2}$$

$$q = \frac{4eV o\alpha\kappa}{md_0^2\,\Omega_T^2}$$

$$\tau = \frac{\Omega_T t}{2}$$

The lowest-order approximation of motion of the ion yields a solution

$$x(t) = x_0\cos(\beta\frac{\Omega_T}{2}t)\left[1-\frac{q}{2}\cos(\Omega_T t)\right]$$

Where $\beta = \sqrt{a+q^2/2}$, and x_0 depends on initial conditions. The slower oscillation at the frequency $\omega_x = \beta\Omega_T/2$, is called the secular motion. When $a << q^2 << 1$ and static

potential $U_0 \cong 0$, then the ion acts as though it is confined in a harmonic pseudo-potential where

$$V = \frac{1}{2} m \omega_x^2 x^2$$

With $\omega_x = \sqrt{2} e V_0 \alpha \kappa / (m d_0^2 \Omega_T)$ being the secular frequency. The secular motion is used as a quantum databus where information can be transferred from one ion to another. The oscillation at the faster frequency $\cos(\Omega_T t)$ is called the micromotion because the amplitude is suppressed by $q/2 = 2\sqrt{2}\omega/\Omega_T$.

The linear trap consists of four rods running parallel Figure 2.1(b) along the *z* axis. Rf is applied to a pair of diagonal electrodes while static potential is applied to the other two electrodes that are segmented. The outer segments are held at U_0 while the middle segments are held at ground. In this case, the potential in a trap becomes

$$V = V_0 \cos \Omega_T t \left(\frac{x^2 - y^2}{r_0^2} \right) + \kappa U_0 \left(\frac{2z^2 - x^2 - y^2}{2} \right)$$

Where κ is a geometric factor. The trap provides a static harmonic potential in the *z* direction, with oscillation frequency $\omega_z = \sqrt{2\kappa U_0 q / m}$. Along the axis of a trap where $x = y = 0$, there is no micromotion. Compared to the ring-and-fork trap where the rf node is a single point at the center of the trap, the linear trap can support many ions at its rf nodal line simultaneously and avoid unwanted micromotion.

Figure 2.2 Vacuum apparatus enclosing the three-layer linear trap. Located inside the 4 inch diameter hemisphere chamber in the lower right corner, the trap is visible from large quartz window. The RF resonator is connected to the trap electrodes by the feedthroughs on top of the chamber. The vertical structure in the middle is the Titanium sublimation pump with the ion pump to its left.

An ion trap is placed under ultra high vacuum inside the hemispheric chamber in Figure 2.2. The chamber has optical access from the cross-section of the hemisphere and two smaller windows at 45^0 around the equator and forming a right angle with each other as shown in Figure 2.3. Trap voltages are controlled from outside and connected to the electrodes via feedthroughs. An rf resonator consisting of a helical coil inside an enclosed cylinder made of copper transforms a 2W rf signal from an amplified source to high voltage and is attached to the trap electrodes via vacuum feedthroughs.

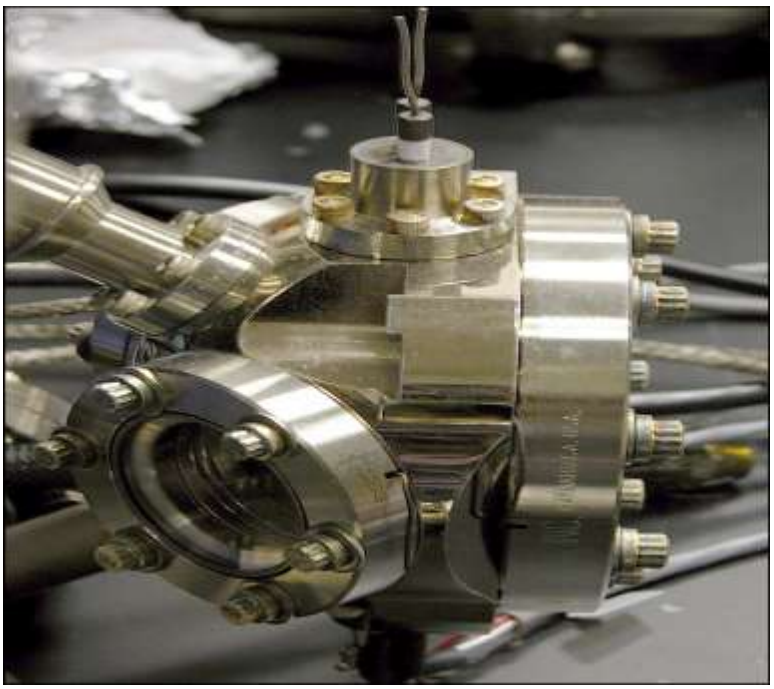


Figure 2.3 A side view of the vacuum chamber. Two windows at 45^{o} allow laser access to the ions inside the trap.

2.2.2. Loading Ions.

The design for loading ions in a trap is as follows: cadmium or cadmium oxide metal inside an alumina tube is heated by passing current through a tungsten coil to produce a beam of atomic vapor aimed in the vicinity of the trap; an electron gun consisting of a charged plate with an aperture in front of a heated tungsten coil generates a beam of electrons aimed at the trap region. The high velocity electrons (~100eV) collide with the neutral cadmium atoms and ionize them, leaving the positively charged cadmium ions inside the trap. If the atoms are ionized outside the trap region, conservation of energy dictates that the ion entering the trap region has higher energy than the trap barrier, therefore these ions will not stay trapped. On the other hand, if the atoms are ionized inside the trap region, then the ions cannot escape as long as the depth of the trap is larger than the initial kinetic energy of the ion. With careful alignment of cadmium oven and

electron gun with respect to the trap electrodes, this method is very effective at loading ions into the trap.

2.2.3. Cadmium 111 as a Qubit.

The type of ion is selected with careful consideration of the technology available and of properties favorable towards quantum computation. Typical atomic ion species for quantum information application are hydrogen-like, with a single valance electron with a $^2S_{1/2}$ ground state. The ion's internal electronic state serves as the quantum memory and two states are designated as the qubit levels |0> and |1>, where information can be stored in the amplitude and phase of these states. A "hyperfine qubit" uses two ground state hyperfine levels, while an "optical qubit" uses the ground state and an excited D state with energy lower than the P state. A "hyperfine qubit" of cadmium ion has the advantage of having an extremely long life, on the order of thousands to millions of years, compared to the life time on the order of seconds for the excited D state. In addition, hyperfine qubits can be manipulated using simulated Raman transition via coupling of excited states, thus having a less stringent requirement on the frequency stability of the laser than optical qubits, which are coupled using electrical quadrupole transitions with linewidths on the order of less than 1 KHz. The experiment in Lee's thesis [PHDLee] uses ground state hyperfine levels of $^{111}Cd^+$ ions as qubits. Hyperfine interaction exists only in isotopes with a non-zero nuclear spin (*I=1/2* for $^{111}Cd^+$). The qubit states are defined as |0> = |F = 0, m_F = 0> state and |1> = |F = 1, m_F = 0> state, where F denotes the angular momentum of electron and nucleus and m_F denotes the *z* component of the total angular momentum. These levels are especially chosen for their

insensitivity to magnetic field, with coherence time of the qubit memory on the order of a few seconds according to Lee's measurements. The frequency splitting is $\omega_o = 2\pi \times$ 14.53GHz.

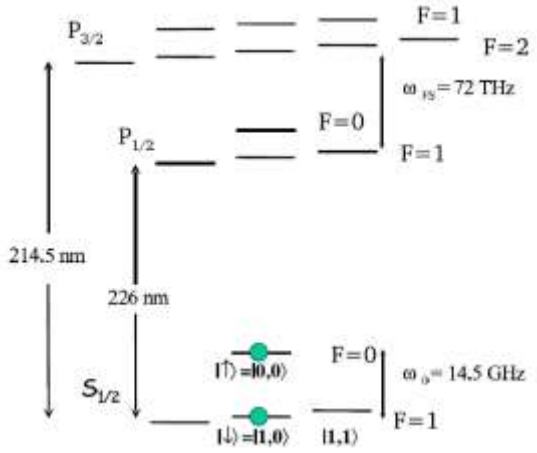


Figure 2.4 Internal energy level of a $^{111}Cd^{+}$ ion. Ground state hyperfine levels (nuclear spin I = ½) serve as a qubit, with a frequency splitting of 14.5GHz. The excited states $P_{1/2}$ and $P_{3/2}$ are separated by a fine structure splitting of 72THz, and the transition from $S_{1/2}$ is resonant with a 214.5nm and a 226nm ultraviolet radiation respectively. The qubit levels |F = 0, m_F = 0> and |F = 1, m_F = 0> states are magnetic filed insensitive to first order, resulting in a coherence time on the order of few seconds.

2.2.4. Initializing and detecting the Qubit states.

A quantum register needs to be initialized to a definite state before any operation can be performed. For trapped $^{111}Cd^{+}$ ions, the qubits are prepared in $S_{1/2}$(F = 0, m_F = 0) ground state with nuclear spin *I=1/*2 at the onset of computation by optical pumping with radiation near-resonant to the $S_{1/2}$(F = 1) → $P_{3/2}$(F' = 1) transition. For this transition, there is always a dark state composed of a superposition of the $S_{1/2}$(F = 1) manifold for polarization of the light. Therefore the optical pumping field is tuned to be in between the $S_{1/2}$(F = 1) → $P_{3/2}$(F' = 1) and $S_{1/2}$(F = 1) → $P_{3/2}$(F' = 2) transition in order to remove any remaining population of state in the dark state. The exact polarization of the laser

light is not critical as long as it is not purely σ^+ or (σ^-) polarized. Preparing the ions in the $S_{1/2}$(F = 0, m_F = 0) state with a 60μW beam focused to a 20 μm waist typically takes about 1 μs based on the laser experimental setup [PHDLee].

Assuming sufficient time is always given to the optical pumping process, the initialized state $S_{1/2}$(F = 0, m_F = 0) still maintains a small probability of coupling off-resonantly to the $P_{3/2}$(F = 1) states and decaying to a $S_{1/2}$(F = 1) state through spontaneous emission. The time for one scatter from $S_{1/2}$(F = 1) to the $P_{3/2}$(F' = 2) state, the probability of the $S_{1/2}$(F = 0, m_F = 0) state leaking to the $S_{1/2}$(F = 1) state from undesired scattering is approximately the square of the detuning ratio (400MHz/14.1GHz) ^{2} = 8 × 10^{-4}. This error is sufficiently small to not significantly affect the fidelity of qubit measurements or operations.

In binary quantum computation a qubit can be in superposition of both |0> and |1> states and/or be entangled with the state of other qubits while quantum operations are performed. However, results of computation can only be obtained by measurements, which collapse the qubit into a |0> or |1> state and potentially alter the state of the entangled system. The entangled system is a fundamental property of quantum systems, in which two quantum systems can become correlated in such a way that the two systems retain this correlation such that, under certain circumstances, subsequent action on one system can then have implications for the outcome of a measurement on the other. In the Cd 111 system, the state of an ion can be detected by applying a σ_ polarized radiation

resonant with the cycling transition $S_{1/2}(\mathrm{F}=1) \rightarrow P_{3/2}(\mathrm{F}=2)$ and collecting the florescence using a CCD camera. The |1> state is optically pumped to the $S_{1/2}(\mathrm{F}=1,\ m_F=-1)$ state and continuously cycled and scattered from the $P_{1/2}(\mathrm{F}=2,\ m_F=2)$ state. Since the other qubit state is far detuned from the only available scattering channel $S_{1/2}(\mathrm{F}=0) \rightarrow P_{3/2}(\mathrm{F}=1)$, the ions in this state remain dark. A photomultiplier tube (PMT) or a camera collects the emitted photons for a certain amount of time. If the emitted photon counts exceed a certain threshold, the ion is determined to be in the one state. Otherwise ion is determined to be in the zero state.

2.2.5. Two ions entangling gate.

The scheme proposed by Cirac and Zoller implements a controlled-NOT (CNOT) through coupling of each qubit to a common mode of motion in the trap. The CNOT gate flips the state of a target qubit (e.g. $|\downarrow_2\rangle \leftrightarrow |\uparrow_2\rangle$) only when the control qubit is in state $|\downarrow_1\rangle$. This universal two qubit logic gate yields the following truth table:

$$|\uparrow\uparrow\rangle \rightarrow |\uparrow\uparrow\rangle$$

$$|\uparrow\downarrow\rangle \rightarrow |\uparrow\downarrow\rangle$$

$$|\downarrow\uparrow\rangle \rightarrow |\downarrow\downarrow\rangle$$

$$|\downarrow\downarrow\rangle \rightarrow |\downarrow\uparrow\rangle$$

How can the spin of one state be rotated coherently conditional on the state of the other qubit? Upon closer examination of the scheme, the CNOT gate can be decomposed into three steps:

(1) a carrier $\pi/2$ pulse on the target qubit with associated phase ϕ

(2) a π phase gate on two ions

(3) a carrier $-\pi/2$ pulse on the target qubit with phase ϕ (step (1) reversed)

Steps (1) and (3) are simply carrier coupling on the target qubit ion, and the step (2) contains the essential entangling process. In the original proposal, the internal qubit state of the control ions is mapped onto the collective motion, and a 2π pulse coupling a specific spin and motion state of the target ion to an auxiliary sate outside of the spin/motion system is applied, resulting in a π phase shift only for that particular state. The motion is then mapped back on the control qubit. The key mechanism here is to obtain a phase that is dependent on the qubits' state, which is satisfied by a resonance condition for accessing the auxiliary level. This π phase gate produces a truth table shown above. The Figure 2.5 shows the fluorescence detected by the CCD camera that allows us to differentiate two ions during detection.

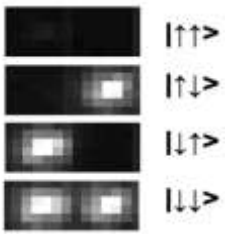


Figure 2.5 Detecting the state of two qubits with an intensified CCD camera. The images corresponds to the states shown in the figure. With CCD camera states can be differentiated which can not be done using PMT (photo multiplier tube).

2.2.6. Quantum charge coupled device (QCCD).

Quantum charge coupled device architecture was proposed [Kielpinski02, Cirac95] to build large-scale quantum computers. An array of ion traps is created in which computation can be performed within each ion trap, and a mechanism is established to facilitate communication between these ion traps. The basic QCCD model is shown in

Figure 2.6a. Ion traps are created between electrode segments that operate on varying voltages to trap or shuttle ions through traps. As shown in Figure 2.6a, the trapped ions storing quantum information are held in the memory region. In order to perform certain logic gate operations relevant ions are moved into interaction regions by applying appropriate voltages to the electrode segments. The required trapping and transport potential are provided using a combination quasi-static and radio-frequency electric field. The quasi-static field is provided by electrode segments and multiple layers are built to facilitate radio-frequency signals and electrode segments. Each ion can be considered as an independent qubit or a cascade of multiple ions can represent single qubit. This type of ion trap region naturally creates regular structure which inevitably lead to constructing FPGA-like regular arrays. The major challenge in accurately operating ion traps is decoherence. Decoherence is a process of disturbing the quantum state through the interaction with the environment which causes inaccurate computational results. Research community working in the area of quantum computing has shown significant interest in addressing the error correction problem, which results from the existence of decoherence. Based on the QCCD model, a Quantum Logic Array (QLA) architecture is proposed in [Metodi05], that takes into account the fundamental requirements for quantum computers.

1. Realizable and realistic implementation technology.
2. Robust error correction and fault tolerant structure.
3. Efficient quantum resource distribution.

The technology used for implementing quantum information processors must satisfy the following four requirements,

1. It must allow initialization of an arbitrary *n*-qubit quantum system to a known state.
2. A universal set of quantum logic operations must be available to manipulate q-bits.
3. The technology must have ability to reliably measure quantum systems.
4. It must allow a sufficiently long qubit lifetime.

The quantum logic gate is a multiple qubit operation, thus it implies that quantum architecture must also allow for sufficient and reliable communication between physical qubits. Due to the high volatility of quantum data, actively stabilizing the system's state through error correction will be one of the most vital operations through the course of implementing quantum algorithms. A successful architecture must be carefully designed to minimize the overhead of error correction, and must be able to accommodate some of the most efficient error correction codes.

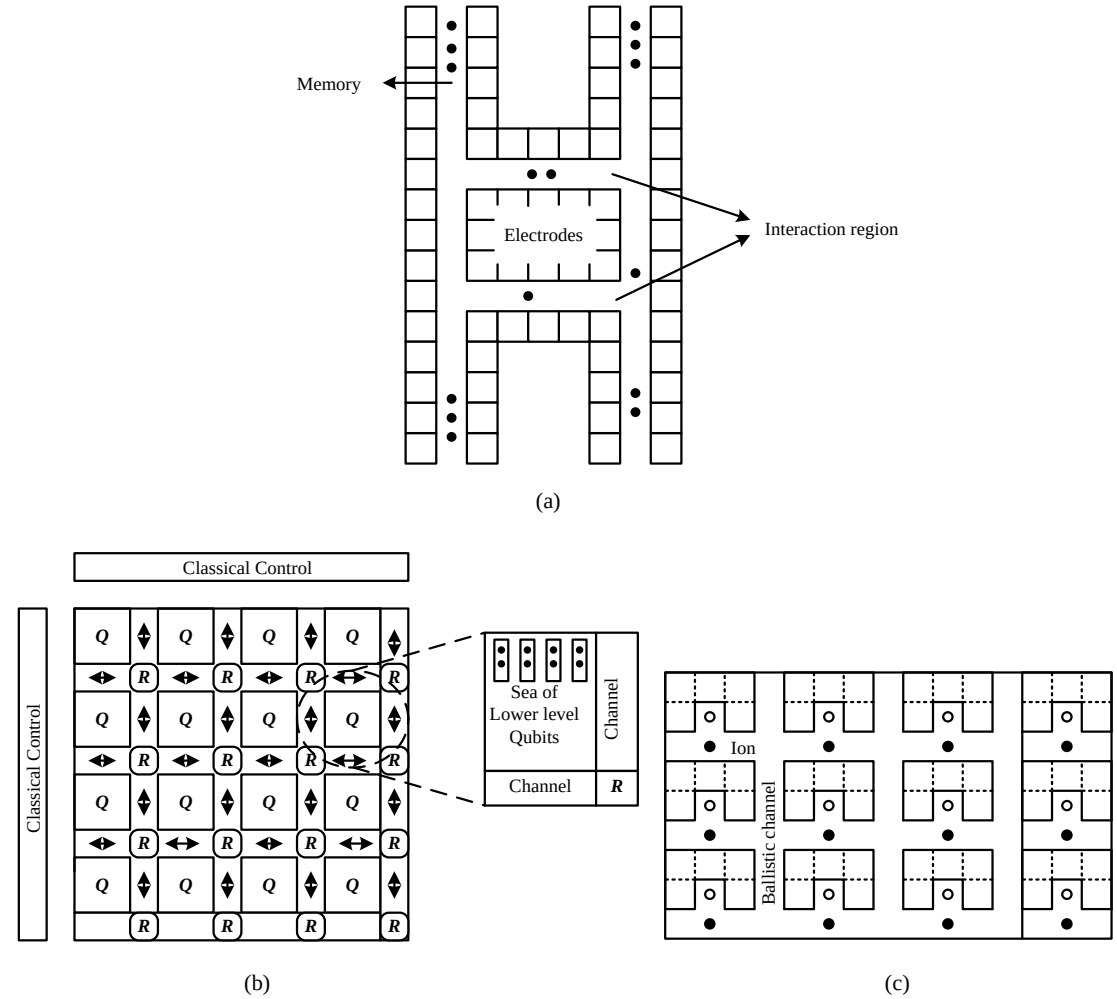


Figure 2.6 Quantum Logic Array architecture, (a) QCCD model in which ions are ballistically shuttled between memory region and interaction regions. (b) High level quantum computer structure where the letters Q represents the qubit and letters R represents switch islands for quantum data communications. (c) The building block of the QLA architecture 4 level 1 building blocks are shown where the far right side outlines a single block. The solid circles are data ions and clear circles are cooling ions.

2.2.7. Quantum Logic Array (QLA) Architecture.

Figure 2.6b shows the QLA architecture proposed by Metodi [Metodi05], which is designed with error correction capabilities. This architecture also shows an efficient method for teleportation, which is used for communication between qubits. At the lowest level QLA is based on trapped ion technology, which uses a single trapped atomic ion as storage for a qubit. In particular, QLA is based on the highly scalable model of (CCD)

style ion-trap quantum information processing architecture proposed by Kielpinski [Kielpinski02]. This model consists of an ion trapped in interconnected trap arrays and moved from trap to trap to interconnect. The QLA is designed as a block structure, which fits naturally for quantum error correction, where each building block reflects the error correction algorithm used. The QLA itself is built by tiling these building blocks to form the hierarchies required for larger and more reliable encodings. In addition, channels are created to facilitate movement of ions from trap to trap.

Figure 2.6b shows the high level structure of a QLA system. The computational units denoted by Q are encoded logical qubits that represent a single qubit of information. Each logical qubit is a regular structure of physical ions controlled by sequences of laser pulses, as shown in Figure 2.6c. The logical qubits are positioned on the substrate in a regular array fashion, connected with tightly integrated repeater-based interconnect as shown in Figure 2.6b. This makes high level design very similar to classical tile based FPGA like architectures. The key difference is that the communication paths must account for data errors in addition to latency. The communication paths are composed of similar physical building blocks as the logical qubits. The integrated repeaters, denoted by the letter R in Figure 2.6b, are called teleportation islands. They redirect traffic in the four cardinal directions by teleporting data from one repeater to the next.

The underlying structure of the QLA is intended for error correction, by far the most dominant and basic operation in the quantum machine. Error correction is expensive because arbitrary reliability is achieved by recursively encoding qubits at the cost of both

exponential resources and operation overhead. Recursive error correction works by encoding *N* physical ion-qubits into a known highly correlated state that can be used to represent a single logical data bit. This data bit is now at level 1 recursion it now has the property of being in a superposition of "0" and "1", much like a single physical qubit. The QLA structure fits naturally with quantum error correction because the structure of the building blocks reflects the error correction algorithm used. Each basic building block represents a single level-one logical qubit as shown in Figure 2.6c. As shown in the figure, each building block consists of data ions supported by their cooling ions and trapped between electrode cells.

2.3. Mathematical Background of Binary Quantum Circuits, qubits and measurements.

Now that the basic Ion Trap architecture has been explained for completeness of the thesis, I move to present the quantum gates that are our real concern in this thesis. It is not required to understand previous sections of this chapter to appreciate the new ideas of this thesis and its contents, but understanding of the next sections is a must with this respect.

2.3.1. Bloch Sphere, rotations and gates.

The most common graphical representation of a qubit uses the concept of a Bloch Sphere. The state of a qubit is represented using three real numbers θ, φ, γ, as

$$|\psi\rangle = e^{i\gamma}\left[\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right] = \alpha_0|0\rangle + \alpha_1|1\rangle$$

With

$$\alpha_0 = e^{i\gamma}\cos\frac{\theta}{2} \qquad\qquad \alpha_1 = e^{i\gamma}\, e^{i\varphi}\sin\frac{\theta}{2}$$

In this representation $e^{i\gamma}$ is an overall phase factor that is not observable in measurements and is generally ignored. Figure 2.7a shows a geometrical representation of the state of the qubit $|\psi\rangle$. The qubit is a vector **r** from the origin to a point on the three-dimensional sphere with a radius of one. In this representation, the position of a point is defined by θ, the angle between vector **r** and the *z* axis, and φ, the angle between the projection of the vector in the *xy* plane and the *x* axis. Figure 2.7b shows the Bloch sphere representation of one qubit in the superposition state $|\psi\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle)/\sqrt{2}$. In this case $\alpha_0 = \alpha_1 = \frac{1}{\sqrt{2}}$ and we can easily derive:

$$\alpha_0 = \cos\frac{\theta}{2} = \frac{1}{\sqrt{2}}, \text{ thus } \theta = 90^o$$

$$\alpha_1 = e^{i\varphi}\sin\frac{\theta}{2} = \frac{1}{\sqrt{2}}, \text{ thus } \varphi = 0^o$$

The classical bit can be in one of two states, 0 or 1, while a qubit can be in a continuum of states represented as points on the Bloch sphere, as shown in Figure 2.7(c). The state space of a qubit contains the two "basis" or "logical" states $|0\rangle$ *and* $|1\rangle$, also called "kets". The initial state of a qubit is always assumed to be one of the basis states. The classical information revealed by a qubit is the label of one of the two basis states, while the contents of the quantum information encoded by a qubit is considerably richer.

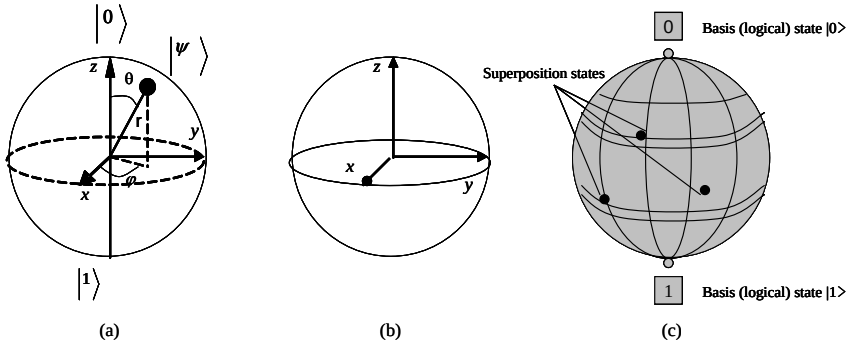


Figure 2.7 Bloch Sphere. (a) A qubit |ψ> is represented as a vector ***r*** *from the origin to the point on the sphere with a radius of one. θ is the angle of the vector* ***r*** *with the z axis and φ is the angle of the projection of vector in xy plane with the x axis. (b) A qubit in the superposition state. (c) A qubit can be in one of the basis states 0 or 1, or in a superposition state.*

A good metaphor to think intuitively about a qubit is to perceive it as a little spin that can rotate in three axes. Operators on this qubit are external forces such as radiation that change the orientations of the spin. The state of the spin changes in the same place, and the changes are executed in time. The "gates" are thus virtually created in time by external forces. The quantum circuit model invented by Deutsch is thus different from classical (CMOS) circuits. In the classical model the gates exist in space and the signal is propagated in them in time. When the change of signal passes the gate, the gate still exists in the circuit. It is not so in a quantum circuit, as the gate changes dynamically to another gate and only its state remains. This is a very different intuition when compared to classical circuits, and, although the quantum circuit drawing looks like a classical circuit schematics, its physical interpretation is very different as the vertical axis X in it is time, not space.

Figure 2.8 presents the symbols and unitary matrices of Pauli rotation gates and the Hadamard gate. Figure 2.9 presents the symbols and unitary matrices of the V gate (short notation for the Square-Root-of-NOT gate), the Phase gate S, the pseudo-Hadamard and the inverse pseudo-Hadamard. These are all single-qubit gates. They are all composed from basic rotations. Ternary or quaternary single qudit gates are also compositions of basic rotations. They differ from binary only in the measurement phase, as a different observable (measurement apparatus) is constructed for them.

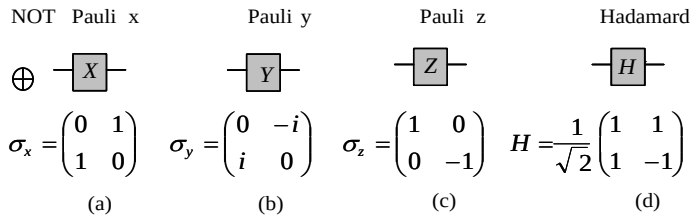


Figure 2.8 Basic single qubit gates, symbols and their respective unitary matrices in Hilbert space. (a) NOT (or Pauli X), (b) Pauli Y, (c) Pauli Z, (d) Hadamard.

TABLE 2.1

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{j\pi}{4}} \end{bmatrix}, \quad P(\phi) = e^{\frac{j\phi}{2}} I, \quad X(\phi) = \cos\frac{\phi}{2} I - j \sin\frac{\phi}{2} X,$$
$$Y(\phi) = \cos\frac{\phi}{2} I - j \sin\frac{\phi}{2} Y, \quad Z(\phi) = \cos\frac{\phi}{2} I - j \sin\frac{\phi}{2} Z$$

It is a good intuition for a circuit designer to think about quantum gates as unitary matrices and compositions of gates as Kronecker and matrix multiplication operators.

Observe that the Hadamard gate is its own inverse, as can be checked by multiplying matrices, but the pseudo-Hadamard gates h, h^{-1} introduced below are not their own inverters. However, multiplying h and h^{-1} gives the identity, which is very useful in several gate realizations and equivalence transformations (Figure 2.9).

Table 2.1 has unitary matrix equations for P, X, Y and Z rotations. These equations use identity matrix I, and rotation matrices X, Y and Z, which are the Pauli rotations from Figure 2.8. Figure 2.14 illustrates how to use these matrix equations to transform these generalized rotations to a form that will be used later in the dissertation. Observe that in rotations P(ϕ), X(ϕ), Y(ϕ) and Z (ϕ), the angles in cosines and sines are actually only ϕ/2, the interpretation of which should also be appreciated on the Bloch sphere and in the future examples given by us. The fact that a single matrix represents a gate says nothing about the complexity of this gate, as every matrix must be first decomposed to matrices of gates realizable in given quantum technology. The unitary matrices of quantum theorists and mathematicians are thus not the unitary matrices of practical quantum circuit designers and quantum CAD engineers!! This is another difference from classical design, where the AND gate is the same at any stage of design.

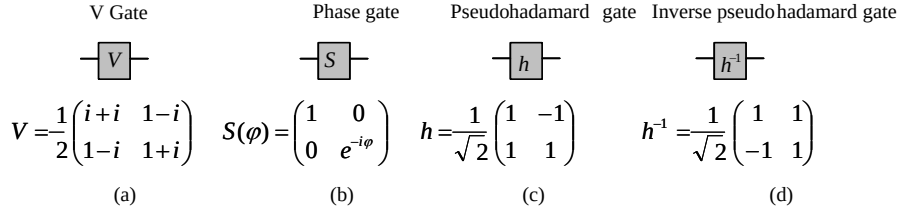


Figure 2.9. (a) Controlled Square Root of NOT (or V gate), (b) Phase Gate, (c) pseudo-Hadamard and (d) inverse pseudo-Hadamard.

Figures 2.10, Figure 2.11, Figure 2.12 and Figure 2.13 have many examples of controlled gates and some of their matrices. The reader can build other matrices using the same principle – Figure 2.10 shows the Identity matrix, and a matrix of controlled gates is shown in Figure 2.11. This means that if the control signal (upper qubit) is zero then the identity operation is executed on the lower qubit. Otherwise, when the control signal is one the operation in the box is executed on the lower input qubit.

Observe also in Figure 2.11 (b) that the CNOT gate (Feynman gate) can be realized with a Controlled-Z gate surrounded by two Hadamard gates, and it also can be realized as in Figure 2.11 (c) where the Hadamard gates are replaced with the pseudo-Hadamard gates and inverse pseudo-Hadamard gates. The pseudo-Hadamard gates are not self-inverses, but $h*h^{-1} = h^{-1} * h = I$, which is very useful in several gate realizations and equivalence transformations. Observe that the circuit from Figure 2.11b has a quantum cost of five pulses, while the same quantum logic is realized with a cost of also three in Figure 2.11c.

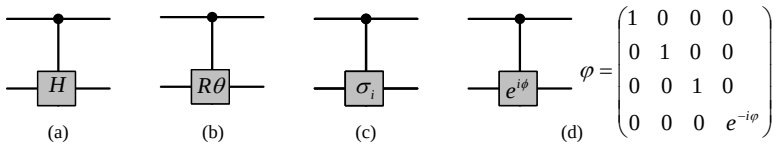


Figure 2.10. Controlled gates. (a) Controlled Hadamard gate, (b) Controlled Rotation with respect to angle θ. This symbol applies to any angle, particularly X, Y and Z. Additional symbol is used to denote the angle, (c) symbol of Pauli rotation where subscript i = X,Y,Z, (d) controlled phase and its unitary matrix.

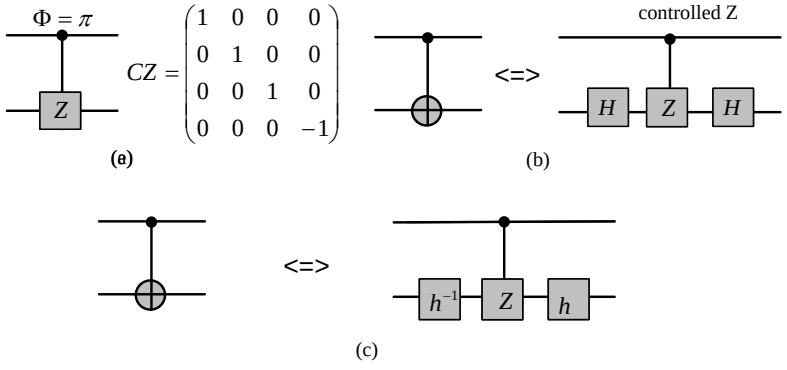


Figure 2.11. (a) Controlled Z and its unitary matrix, (b) CNOT realized with controlled Z and Hadamard gates, (c) CNOT realized with controlled-Z and pseudo-hadamard gates. Symbol h stands for pseudo-hadamard gate and symbol h^{-1} for inverse pseudo-hadamard gate.

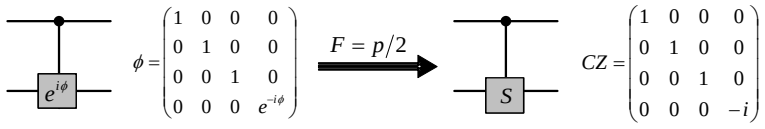


Figure 2.12. Realization on controlled phase gate.

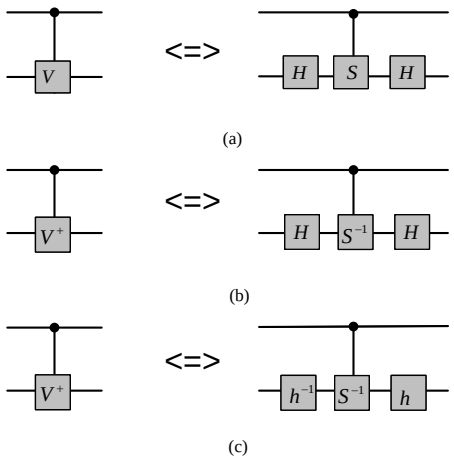


Figure 2.13. (a), Controlled V gate and its equivalent realization with controlled phase gate surrounded by Hadamard gates, (b) Controlled V^+ gate and its equivalent realization with inverse of phase gate surrounded by Hadamard gates, (c) Controlled V^+ realization with inverse of phase gate surrounded by inverse pseudo-hadamard gate and pseudo-hadamard gate.

Similarly the controlled-square root of NOT (controlled V gate) gate implementation using a phase gate surrounded Hadamard gates is shown in Figure 2.13 (a). Figure 2.13 (b) depicts controlled V adjoint gate with inverse of phase gate and Hadamard gate and Figure 2.13 (c) shows implementation of same gate with inverse of phase gate surrounded by pseudo- Hadamard and inverse pseudo-Hadamard gate. We can appreciate again that using h and h^{-1} reduces the cost of the gate CV, CV^+ which are the most important primitives for building binary quantum oracles. Interestingly, quantum computers have been practically built for more than 10 years and every year new design tricks and rules are found for the design of gates, so I can have a hope that I may invent some useful

concept, such as the quantum multiplexer that was invented by Dr. Perkowski in the year 2000 and the pseudo-Hadamard gate by Prof. Jones in 1998.

A quantum circuit can be easily analyzed. A parallel connection of gates corresponds to the Kronecker product (Tensor product) of the unitary matrices of respective gates. A serial connection of gates corresponds to a serial multiplication (in reverse order) of the matrices of these gates. It is thus easy to check that the equivalence transformations from Figures 2.11, Figure 2.12 and Figure 2.13 are correct. Figure 2.12 presents the controlled general phase gate used together with a pair of pseudo-Hadamard and its inverse. Figure 2.12(a) has the symbolic unitary matrix when the control signal is $|1\rangle$. By substituting various values of angles, 0^o, 90^o, -90^o, 180^o unitary matrices are created which are next combined with pseudo-Hadamard matrices as in Figure 2.18b. The table from Figure 2.18a demonstrates that by changing the angle in the controlled middle gate the gate from Figure 2.13 can work as a 2-qubit identity, controlled-V, controlled-V^+ and CNOT. Actually this gate can be used as controlled root of various degrees. Figure 2.18a illustrates unitary matrices for various angles of Y. This figure thus demonstrates the usefulness of Y and Z rotations to create quantum gates. I will be using single qubit gates that are commonly used in papers on quantum synthesis. They are: the NOT gate (called also the Pauli rotation X, denoted also in literature by σ_x), the Hadamard gate, the Phase gate, and the T gate. Some of these gates are shown in Figure 2.8. Some gates are also shown in Table 2.1. I use Pauli rotations X, Y and Z or arbitrary angle rotations with respect to axes X, Y and Z of the Bloch sphere and some of their special cases for fixed angles which are multiples of 45^o. I will also use two new gates, pseudo-Hadamard *h* and

its adjoint pseudo-Hadamard gate h^{-1} because they are used to build many quantum gates, both the permutative (pseudo-binary) gates and the general-purpose-quantum gates that are most useful in synthesis [Jones98a]. Useful transformations are shown in Figures 2.14 - 2.17.

In Table 2.1 symbols X, Y, and Z are the earlier defined Pauli spin matrices, and P(ϕ), X(ϕ), Y(ϕ), and Z(ϕ) are the corresponding 2×2 matrices of arbitrary parameterized angle rotations by angle ϕ. The rotations X(ϕ), Y(ϕ), and Z(ϕ) can be explained as rotations with respect to angles X, Y and Z giving on the Bloch sphere [Nielsen00]. P is a phase rotation by ϕ/2 to help match identities automatically, its idea comes from [Lomont03].

$$
\begin{aligned}
Y(\phi) &= \cos\frac{\phi}{2} I - i \sin\frac{\phi}{2} Y \\
&= \cos\frac{\phi}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - i\sin\frac{\phi}{2}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\
&= \begin{bmatrix} \cos\frac{\phi}{2} & 0 \\ 0 & \cos\frac{\phi}{2} \end{bmatrix} - \begin{bmatrix} 0 & -i^2\sin\frac{\phi}{2} \\ i^2\sin\frac{\phi}{2} & 0 \end{bmatrix} \\
&= \begin{bmatrix} \cos\frac{\phi}{2} & -\sin\frac{\phi}{2} \\ \sin\frac{\phi}{2} & \cos\frac{\phi}{2} \end{bmatrix}
\end{aligned}
$$

Figure 2.14. Example how to calculate unitary matrices of generalized rotations from general matrix formulas in Table 2.1.

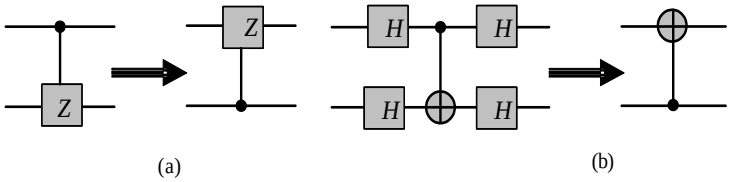
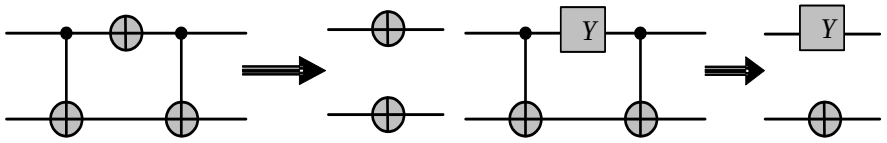
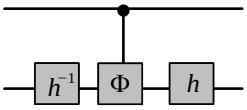


Figure 2.15. (a) Equivalent transformation of Z gate, (b) equivalent transformation of CNOT and Hadamard gates.



(a) (b)

Figure 2.16. (a) CNOT and NOT transformation, (b) CNOTs and Pauli Y transformation.



$$h * \phi * h^{-1} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix} * \begin{bmatrix}1 & 0\\ 0 & e^{j\phi}\end{bmatrix} * \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix}1+e^{j\phi} & 1-e^{j\phi}\\ 1-e^{j\phi} & 1+e^{j\phi}\end{bmatrix}$$

(a) (b)

Figure 2.17. (a) Controlled-Z gate surrounded by pseudo-hadamards, (b) Calculation of unitary matrix for the target qubit of this gate.

$\phi = 0$	I
$\phi = 90^0$	V
$\phi = -90^0$	V^+
$\phi = 180^0$	X

(a)

$\phi = 0$	I
$\phi = 90^0$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix} = h$
$\phi = -90^0$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix} = h^{-1}$
$\phi = 180^0$	$\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}$

(b)

Figure 2.18. (a) Various gates realized by ϕ for angles 0 ^{o}, 90 ^{o}, -90 ^{o} and 180^o in X rotations. (b) gates realized by Y rotations.

2.3.2. Basic operations of Square-Root-Of-NOT and its adjoint gates.

Cascading two Square-Root-of-NOT gates operates as an inverter gate (Figure 2.19). Cascading a Square-Root-of-NOT gate and its adjoint gate works as an identity ($V * V^+ = V^+ * V = I$).

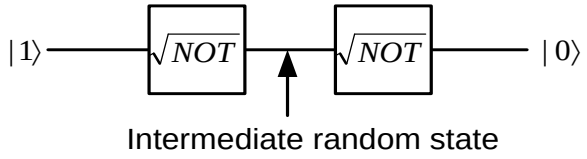


Figure 2.19. Explanation of operation of the Square-Root-of-NOT gate.

A qubit 0 given at the input to the Square-Root-of-NOT gate Figure 2.20 (a) produces output $V_0 = ½ [1+i\ 1-i]^T$. Similarly, all possible output values can be calculated for all possible values of the algebra with set {0, 1, V_0, V_1} of values (Figure 2.20 (b), (c), (d), (e), (f), (g) and (h)).

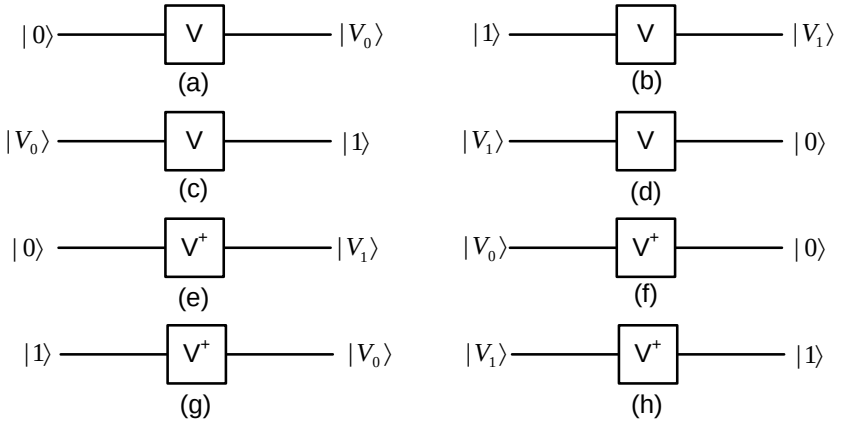


Figure 2.20. Operation of gates V and V^+ on binary states $|0\rangle$ and $|1\rangle$ and quantum states $|V_0\rangle$ and $|V_1\rangle$.

Observe that no new values are created and the set {0, 1, V_0, V_1} is closed; thus we deal here with a four-valued algebra (multi-valued logic). Gates V, V^+ and NOT are rotations and are used to realize binary and quaternary quantum circuits. In order to be universal, we have, however, to be able to control them and build arbitrary-sized gates by multiple controls. These constructions are fundamental to generating universal multi-qudit gates in any radix. In this dissertation I will concentrate on binary (radix 2), ternary (radix 3) and quaternary (radix 4) gates. I will show a unified approach to all these radices. The set of

useful classical gates and their quantum binary equivalent gates are given in Figure 2.21.

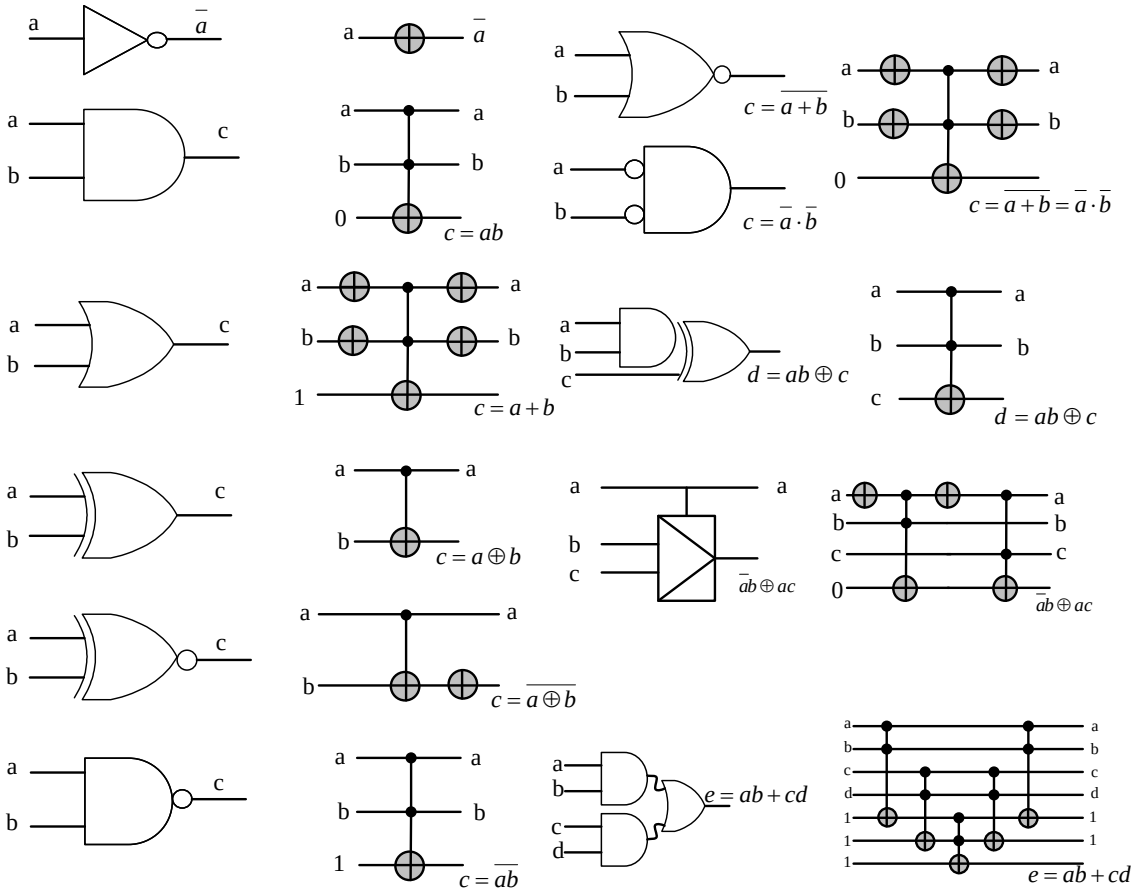


Figure 2.21. Representation of one qubit, two qubit, three qubit and four qubit quantum binary gates and their classical counterpart gates. Observe the use of undesirable ancilla qubits in most gates.

2.4. Mathematical Background of Ternary Quantum Logic.

2. 4.1. Qudits and Dirac Notation for ternary quantum circuits.

In the ternary quantum logic system, the unit of memory (information) is a qutrit. Ternary logic values of 0, 1, and 2 are represented by a set of distinguishable different quantum states of an object that represents the qudit. After encoding these distinguishable quantities into ternary constants, qutrit states are represented by $|0\rangle$, $|1\rangle$, and $|2\rangle$,

respectively, and are called the computational basis states. Qutrits exist in a linear superposition of basis states, and are characterized by a wave-function ψ. In ternary quantum logic, the notation for the superposition is $\alpha|0\rangle+\beta|1\rangle+\gamma|2\rangle$, also written as a vector (α,β,γ), where α, β, and γ are complex numbers. These intermediate states cannot be distinguished, rather a measurement will yield that the qutrit is in one of the basis states, $|0\rangle$, $|1\rangle$, or $|2\rangle$. The probability that a measurement of a qutrit yields state $|0\rangle$ is $|\alpha|^2$ where $|\alpha|^2=\alpha\bullet\alpha^*$ ($\alpha=a+jb, \alpha^*=a-jb, j=\sqrt{-1}$), state $|1\rangle$ is $|\beta|^2$, and state $|2\rangle$ is $|\gamma|^2$. The sum of these probabilities is one, which implies $|\alpha|^2+|\beta|^2+|\gamma|^2=1$. The absolute values are required since, in general, α, β and γ are complex quantities. Once the measurement is done and the result of $|0\rangle$, $|1\rangle$, or $|2\rangle$ is read, every subsequent measurement gives the same result as the one read before. Thus, the meaningful measurement of a single qutrit can be done only once. Afterwards this qutrit should be initialized again. Quantum gates carry around and manipulate quantum information. Any transformation of the qutrit state represented by a 3×3 unitary matrix specifies a valid 1-qutrit (ternary) quantum gate. There are many such non-trivial 1-qutrit gates. The output qutrit state of a gate is determined by matrix multiplication of the unitary matrix characterizing the gate and the vector representing the input qutrit state.

An *n*-qutrit quantum system has 3^n computational basis states denoted as $|00\cdots0\rangle,|00\cdots1\rangle,\cdots,|11\cdots1\rangle$. An *n*-qutrit system exists in superposition of these 3^n states. For example, a 2-qudit ternary system represents nine distinct states,$|00\rangle$, $|01\rangle$, $|02\rangle$, $|10\rangle$,

$|11\rangle$, $|12\rangle$, $|20\rangle$, $|21\rangle$, and $|22\rangle$, as well as all possible superposition of these states. This property may be mathematically described using the Kronecker product (tensor product) operation ⊗ [Nielsen00]. As an example, consider two qudits with $\psi_1 = \alpha_1|0\rangle + \beta_1|1\rangle + \gamma_1|2\rangle$ and $\psi_2 = \alpha_2|0\rangle + \beta_2|1\rangle + \gamma_2|2\rangle$. When the two qutrits are considered to represent a state, that state ψ_{12} is the superposition of all possible combinations of the original qutrits, where

$$\begin{aligned}\psi_{12} &= \psi_1 \otimes \psi_2 \\ &= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \alpha_1\gamma_2|02\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle + \beta_1\gamma_2|12\rangle + \gamma_1\alpha_2|20\rangle + \gamma_1\beta_2|21\rangle + \gamma_1\gamma_2|22\rangle\end{aligned}$$

Any $3^n \times 3^n$ unitary matrix represents an *n*-qutrit gate. Determining the output qutrit state of such *n*-qutrit gates requires matrix multiplication of one $3^n \times 3^n$ matrix characterizing the gate and one $3^n \times 1$ matrix representing the input qutrit state. Currently it is still difficult to build multiple-valued quantum systems with $n > 2$. In this dissertation, we concentrate on the Ion Trap realizable 2-qutrit Muthukrishnan-Stroud gates [Muthurkrishnan00]. Other *n*-qudit ($n \geq 2$) gates such as ternary Feynman and Toffoli gates can be built as macro-level gates on top of ternary 1-qutrit and Muthukrishnan-Stroud gates. New gates and their structures are an innovation of this dissertation.

2.4.2. Potentially useful Operators.

In creating ternary quantum logic, I should take into account both the existing Ion Trap technology and the existing algebras and the ternary-logic synthesis methods. Otherwise I would end up with non-realizable circuits. There are two existing candidates for Toffoli-like generalizations. One is based on Galois Fields.

2.4.3. Galois Field 3 Logic.

The Galois Field 3 (GF3) consists of the set of elements $T = \{0, 1, 2\}$ and two basic binary operations – *addition* (denoted by $\oplus$) and *multiplication* (denoted by $\cdot$ or absence of any operator) as defined in Table 2.2. Readers should note that GF3 operations are nothing but modulo 3 operations.

Table 2.2 Galois Field 3 operations

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\bullet$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

A GF3 variable *A* may have many literal representations [Bullock05, Greentree04], but in this dissertation, I will start first only with the normal variable *A*.

2.4.4. Ternary 1-qutrit permutative gates.

Any transformation of the qutrit state represented by a 3×3 unitary matrix specifies a valid 1-qutrit ternary quantum gate. There are many such non-trivial 1-qutrit gates. However, in this work, I use only the permutative transforms as shown by permutative matrices of Figure 2.22 (see [Perkowski02] and [Khan04]). Transforms $Z_3(+1)$ and $Z_3(+2)$ shift the qutrit states by 1 and 2, respectively. Transform $Z_3(12)$ permutes the qutrit states $|1\rangle$ and $|2\rangle$, $Z_3(01)$ permutes the qutrit states $|0\rangle$ and $|1\rangle$, and $Z_3(02)$ permutes the qutrit states $|0\rangle$ and $|2\rangle$ without affecting the other qutrit state. The logical equivalent of these 1-qutrit transforms can be expressed using GF3 expressions and truth tables as shown in

Table 2.3. Observe that these are the permutative ternary operations that directly realize the binary rotations from Section 2.3.2. They can be realized in terms of single qubit gates from section 2.3.3. These permutative operations are thus good both technologically and algebraically.

The transforms $Z_3(+1)$ and $Z_3(02)$ are referred to as *C*1 and *N*, respectively, by De Vos *et al.* [DeVos02] and the same notations are used by Miller *et al.* [Miller04, Miller04a]. The transforms $Z_3(+2)$, $Z_3(12)$, and $Z_3(01)$ are referred to as *C*2, *D*, and *E*, respectively, in [Miller04, Miller04a].

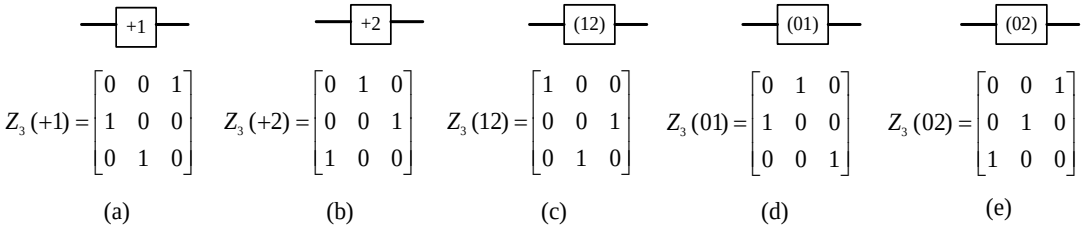


Figure 2.22. 1-qutrit ternary permutative transforms and symbolic representation.

Table 2.3 1-qutrit ternary permutative transforms.

A	$A(+1) = A+1$	$A(+2) = A+2$	$A(12) = 2A$	$A(01) = 2A+1$	$A(02) = 2A+2$
0	1	2	0	1	2
1	2	0	2	0	1
2	0	1	1	2	0

In [Miller04a], the quantum implementation technology is assumed to be (liquid) NMR, and it is stated that the 1-qutrit *C*1 gate is quantum rotation gate $R(2\pi/3)$, *C*2 is rotation $R(-2\pi/3)$, and *N* is $-Z$ (where *Z* is the Pauli-*Z* gate). These gates are elementary building

blocks and their gate costs are assumed to be 1. In Ion Trap technology, the 1-qutrit gates $Z_3(+1)$, $Z_3(+2)$, $Z_3(12)$, $Z_3(01)$, and $Z_3(02)$ can be implemented directly as elementary operations [Muthurkrishnan00]. Using reasoning similar to [Miller04a], I assign the gate costs of these 1-qutrit gates to be 1. If two 1-qudit gates *x* and *y* in cascade have the resultant effect that the input signal to gate *x* is restored at the output of gate *y*, then gate *y* is said to be the *inverse gate* of gate *x*. Table 2.4 shows the inverse gates of all the 1-qudit gates.

Table 2.4. 1-qudit ternary gates and their inverse gates.

Gate	$Z(+1)$	$Z(+2)$	$Z(12)$	$Z(01)$	$Z(02)$
Inverse gate	$Z(+2)$	$Z(+1)$	$Z(12)$	$Z(01)$	$Z(02)$

2.4.5. Chrestenson gate as a generalization of Hadamard with different properties.

The Hadamard gate is very fundamental in quantum computing in general, but it is useless for creating single qubit gates in permutative binary quantum circuits because H*H = I, so inverters cannot be built from it. As we will see below, this is not true when we go to a higher dimension – the generalization of the Hadamard gate, called the Chrestenson gate will be sufficient to create a universal system of gates. As we remember, the Hadamard gate is based on the second order root of unity. Thus by analogy we take third order root of unity in case of ternary logic.

The complex third order root of unity is

$$a = e^{i\frac{2\Pi}{3}} = a\cos\frac{-2\Pi}{3} + ia\sin\frac{-2\Pi}{3} = -0.5 - i\ \ 0.866$$

Observe the powerful analogy. In binary quantum logic to negate we rotate by $2\prod / 2$, while in ternary logic we rotate by $2\prod / 3$. Thus whenever in formulas we have a value 2 in binary, we should expect value 3 in ternary. I used this heuristic in many guesses of new ternary gates and I was often successful in inventing gates. Using the root of unity, the Chrestenson gate performs the mapping in H_3 described by unitary matrices.

$$CH = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix}$$

Unitarity of a gate is easy to check from the definition by multiplying the matrix of this gate by its Hermitian conjugate (conjugate transpose). The result should be an identity matrix.

$$CH * CH^+ = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix} * \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix}^+$$

$$= \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix} * \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & (a)^* & (a^2)^* \\ 1 & (a^2)^* & (a)^* \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & (a)^* & (a^2)^* \\ 1 & (a^2)^* & (a)^* \end{bmatrix}$$

The third root of unity satisfies the property $a^* = a^2$ from which we can derive $a = (a^*)^* = (a^2)^*$

This leads to a product:

$$CH * CH^+ = \frac{1}{3}\begin{bmatrix} 3 & 1+a+a^2 & 1+a+a^2 \\ 1+a+a^2 & 1+a^3+a^3 & 1+a+a^2 \\ 1+a+a^2 & 1+a+a^2 & 1+a^3+a^3 \end{bmatrix}$$

After applying identities $1+a+a^2 = 0$ and $a^3 = 1$ the matrix becomes an identity. As mentioned, I will try to use all kinds of analogies between binary and ternary to be able to reuse as much as possible from the binary methods and properties, provided that the newly created ternary gates are quantum realizable. Thus the ideas of constructing more complex gates from single-qubit gates should also be generalized from binary to ternary. A well-known construction of three-qubit gates from only two-qubit gates comes from Sleator and Weinfurter [Sleator94]. A similar scheme to create two-qubit gates from CNOT and one qubit gates comes from [Barenco95]. The question is, how difficult is to apply these ideas to ternary? To create a 3*3 unitary matrix the problem is reduced to that of finding the unitary operators $U3$, $U2$ and $U1$ such that $U_3\ U_2\ U_1\ U = I$ and thus $U = U_1^{+} U_2^{+}\ U_3^{+} U$. There exists a closed expression for each of the matrices [Nielsen00].

2.5. Two-Qutrit gates.

2.5.1. Ternary Muthukrishnan-Stroud gates.

Muthukrishnan and Stroud [Muthurkrishnan00] proposed a family of 2-qudit multiple-valued gates and showed their Ion Trap realization. The family of such 2-qudit ternary gates can be expressed as acting on a 9-dimensional basis of two qudits, where Y_3 represents two families of 3-dimensional transforms, Z_3, or X_3. $\Gamma_2[Y_3]$ transforms the second qudit by Y_3, conditional on the first qudit being in $|2\rangle$. In this dissertation, I am concerned only with the permutative transform Z_3 as shown in Figure 2.22, that maps a known single-qutrit state to $|2\rangle$, $Z_3(c_0, c_1, c_2) : c_0|0\rangle + c_1|1\rangle + c_2|2\rangle \mapsto |2\rangle$.

We assign the $\Gamma_2[Y_3]$ symbol to the Muthukrishnan-Stroud (M-S) gate, and represent that by the diagram from Figure 2.23.

$$\Gamma_2[Y_3] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & y_{11} & y_{12} & y_{13} \\ 0 & 0 & 0 & 0 & 0 & 0 & y_{21} & y_{22} & y_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & y_{31} & y_{32} & y_{33} \end{bmatrix}$$

Figure 2.22. Unitary matrix for Muthukrishnan-Stroud gate.

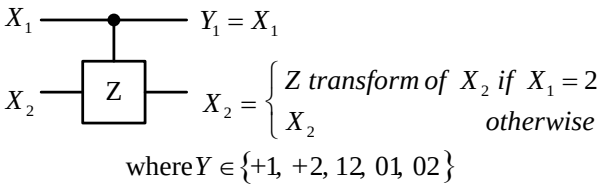


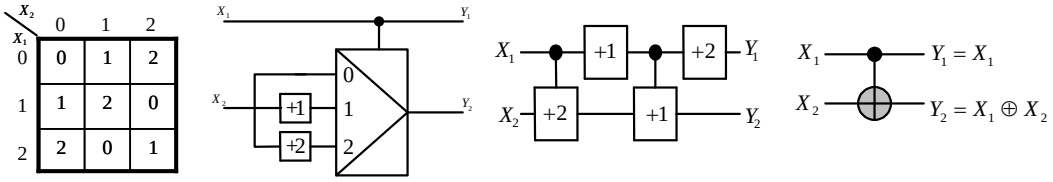
Figure 2.23. Symbol of ternary Muthukrishnan-Stroud gate.

The M-S gates are similar to the controlled 2-qutrit De Vos gates [DeVos02] and their extensions used by Miller *et al.* [Miller04, Miller04a], namely the *CC*1, *CN*, *CC*2, *CD*, and *CE* gates. The only exception is that in the M-S gates the controlling value is 2 and in the De Vos gates (including the extensions) the controlling value is 1. In [Miller04a], the controlled 2-qutrit gates *CC*1, *CN*, and *CC*2 are considered as elementary gates and their cost is assumed to be 1. Using similar reasoning, I assign the gate cost of M-S gates to be one.

2.5.2. Ternary Feynman gate.

The ternary counterpart of the binary Feynman gate is shown in Figure 2.24(d), and its realization is shown in Figure 2.24(b, c). In Figure 2.24(b), when $X_1 = 0$, the data input 0 of the quantum multiplexer is selected, which is qutrit X_2. This corresponds in Figure

2.24(c) to the situation where none of the controlling values of the M-S gates is 2 and no transformation will be applied on X_2. Here, symbol $\oplus$ denotes modulo-3 addition. Therefore, Y_2 will be $X_1 \oplus X_2 = 0 \oplus X_2$. If $X_1 = 1$, then the controlling value of only the second M-S gate will be 2 and Y_2 will be $X_2 \oplus 1 = X_1 \oplus X_2$. If $X_1 = 2$, then the controlling value of only the first M-S gate will be 2 and Y_2 will be $X_2 \oplus 2 = X_1 \oplus X_2$. The right-most 1-qudit +2 gate is the inverse gate of the +1 gate used to restore the value of X_1.



(a) (b) (c) (d)

Figure 2.24. Ternary 2-qudit Feynman gate. (a) Ternary map for qutrit Y_2, (b) realization of Y_2 using the quantum multiplexer formalism, (c) Circuit with ternary M-S gates that corresponds to the quantum multiplexer circuit from Figure 2.24(b), (d) the symbol schematic of the ternary Feynman gate. Modulo-3 addition is used. The symbol is the same as in binary logic so the reader has to observe the context.

Considering the cost of the ternary 1-qutrit and M-S gates to be 1, the gate cost of the ternary Feynman gate implementation of Figure 2.24(c) is 4. As far as I know, there is no published literature showing that the ternary Feynman gate can be implemented as an elementary operation in NMR or Ion Trap, or other technologies. Even so, in [Miller04a] the gate cost of a ternary Feynman gate is assumed to be 1 (see related discussion of [Miller04a]). This is an unrealistic assumption. Moreover, in [Miller04a] the second Feynman gate is the inverse gate of the first Feynman gate to restore the second controlling signal. In the case of binary gates it is true, since in GF2 $A \oplus A = 0$. But in the case of ternary gates, the figure must have two such inverse Feynman gates to restore the

second controlling signal, since in GF3 $A \oplus A \oplus A = 0$. In this respect the cost calculation of 3-qudit controlled gates in [Miller04a] is not realistic. All calculations in this dissertation will be executed with precise quantum costs. This has been done so far only for a few gates and only for binary gates. The ternary case is more complicated.

2.6. Ternary gates with 3 or more qutrits.

2.6.1. Ternary Toffoli gates.

A 3-qutrit ternary Toffolli gate is shown in Figure 2.25(a), where X_1 and X_2 are two controlling inputs and X_3 is the controlled input. If the two controlling input values are 2, then the *Z* transform is applied on the controlled input, otherwise the controlled input is passed unchanged. Realization of this gate using M-S gates is shown in Figure 2.25(b), where a constant input 0 is changed to 2 by using two +1 transforms controlled from the two controlling inputs X_1 and X_2, and then the resulting constant 2 is used to control the input X_3. The right-most two gates are the inverse gates of the left-most two gates used to restore the constant input 0. The cost of this realization is five, and needs one ancilla bit.

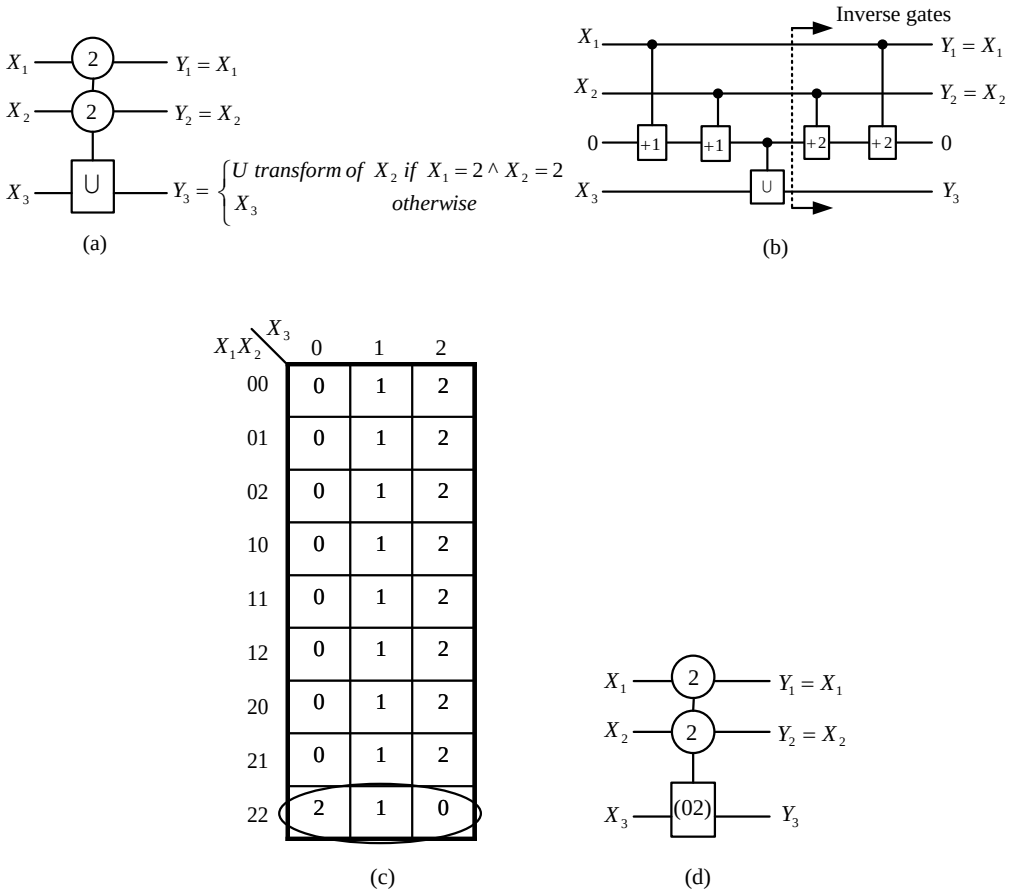


Figure 2.25. 3-qutrit ternary Toffoli gate (a) the general schematic, (b) its realization using ternary M-S gates (c) the ternary map of double-controlled (02) operation, (d) the schematic of double-controlled (02) gate – this is like one of Toffoli-like gate in ternary logic.

A 4-qutrit ternary Toffoli gate and its realization using M-S gates are shown in Figure 2.26. In Figure 2.26(b), controlling inputs X_1 and X_2 when in state 2, add one each to qutrit B, thus together they change the constant input 0 to 2 and that 2 becomes the controlling value B of the +1 gate in qutrit C. Adding two ones in qutrit C creates value 2. This final value 2 is then used to execute operator U on X_3. The cost of this realization is none and needs two ancilla bits. Using a similar technique, any *n*-qutrit ($n > 4$) Toffoli gate can be realized.

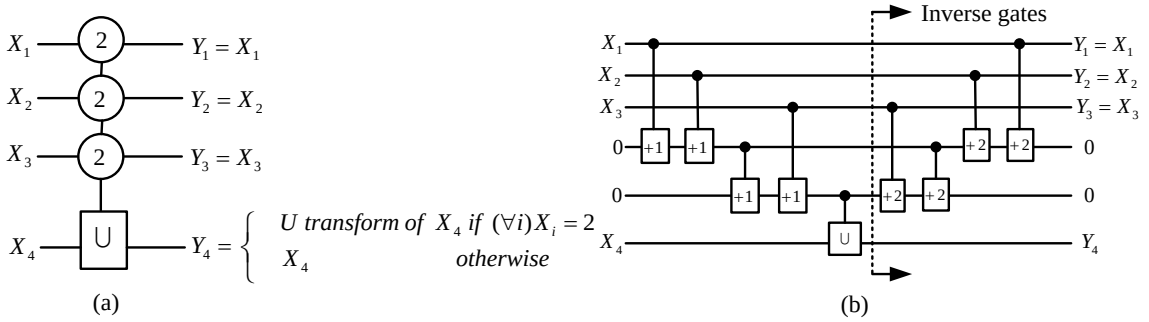


Figure 2.26. 4-qutrit ternary Toffoli gate. (a) symbolic schematic, (b) its realization with M-S ternary gates and two ancilla qutrits. Please analyze the role of the mirror circuit at the right that restores constants 0 in two qutrits, allowing them to be reused in the next gates of this cascade.

A 3-qutrit generalized ternary Toffoli gate is shown in Figure 2.27(a), where X_1 and X_2 are two controlling inputs having controlling values $x_1, x_2 \in \{0,1,2\}$, and X_3 is the controlled input. If the controlling inputs $X_1 = x_1$ and $X_2 = x_2$ then the *Z* transform is applied to the controlled input X_3, otherwise the value of X_3, is passed unchanged. The realization of this gate is shown in Figure 2.27(b), where the controlling value x_i (where $i = 1,2$) is changed to 2 by a 1-qudit gate $+a_i = +(2 - x_i)$ (a +0 represents a quantum wire) and then these two 2s are used to control the input X_3 as is done in the case of Figure 2.25. The inverse gate $+a_i'$ is used to restore the controlling value. The cost of this realization is 5 + 2 × (number of non-2 controlling values). As in Figure 2.25(b), this realization needs one ancilla bit. An *n*-qutrit generalized ternary Toffoli gates can be realized in a similar way, where the cost of the realization will be (cost of *n*-qutrit Toffoli gate) + 2 × (number of non-2 controlling values). Miller *et al.* used similar Toffoli gates in [Miller04, Miller04a]. In [Miller04a], the cost of such 3-qutrit and 4-qutrit controlled gates is stated to be 5 + (2 × number of non-1 controlling values) and 13 + (2 × number of

non-1 controlling values), respectively. As realization details of these gates are not explicitly given in [Miller04a], there is no way of determining the number of ancilla bits required for these realizations.

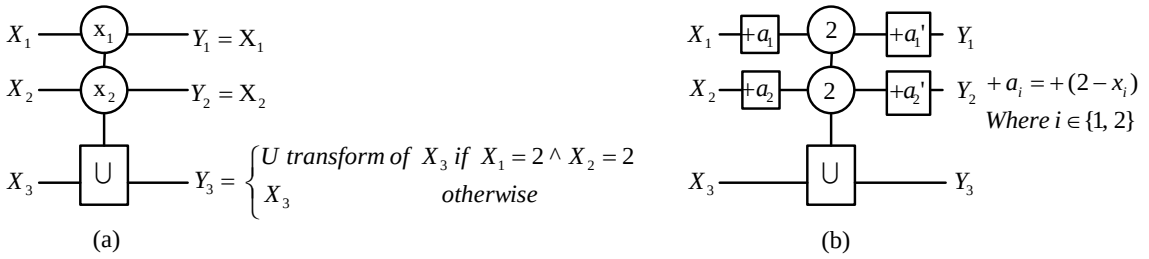


Figure 2.27. A 3-qutrit generalized ternary Toffoli gate with arbitrary controlling values Operator U is executed only if the first qudit X_1 has value x1 and the second qudit X_2 has value x2. (a) general schematic, (b) its realization with the schematic of the ternary Toffoli controlled with values 2 from Figure 2.25(a).

2.6.2. More 3-qutrit gates.

Figure 2.28(a) presents the standard binary Fredkin gate. Please observe that if the control signal = 0 then P = x and Q = y so the gate is an identity. If the control = 1, then the gate swaps, which means that P = y and Q = x. The operation of the ternary Fredkin gate is analogous. When the control = 0 or 2, the gate works as identity. When the control = 1, the gate swaps. Please observe that the signals that are swapped are ternary now. In this variant I discussed the case that 1 is an active control value. Similarly a Fredkin gate can be designed for which value 2 is the active control value, as in MS gates. We can always design gates for both these control variants. Figure 2.28(b) presents the binary SWAP gate realized from Feynman gates. Figure 2.28(c) presents the binary "double SWAP" (permutation) gate and its realization. Figure 2.28(d) presents the realization of the binary controlled double SWAP gate. A imilar gate can be built in ternary using M-S gates.

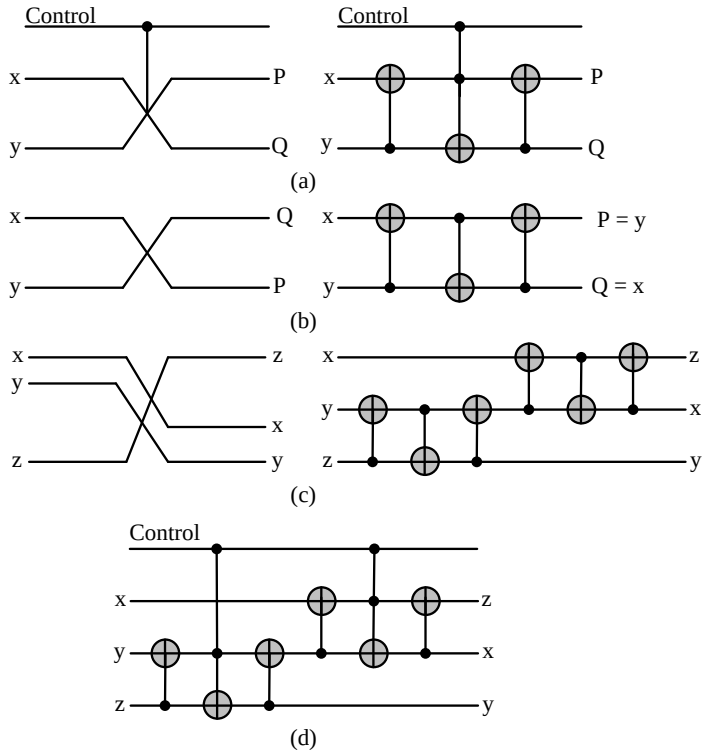


Figure 2.28. (a) Fredkin gate (controlled SWAP) (b) SWAP gate (c) Double SWAP gate (d) double controlled SWAP gate.

2.6.3. Realization of ternary Sum of Minterms using generalized Toffoli gates.

Let X_1, X_2, X_3 be three ternary input variables and $F(X_1, X_2, X_3)$ be the ternary function output. A minterm $x_1x_2x_3$ (where $x_i \in \{0,1,2\}$) producing an output f (where $f \in \{1,2\}$) can be realized using a 4-qutrit generalized Toffoli gate as shown in Figure 2.29. The realization is trivial. A sum of minterms can be realized by cascading the realizations of the minterms. Table 2.5 shows the truth table of a ternary half-adder. For the carry output *C*, the minterms 12, 21, and 22 produce output 1 and the remaining minterms produce output 0. Therefore, the *C* output can be realized as shown in Figure 2.30. When the inputs $X_1X_2 = 12$, then the left-most +1 transform is applied on the input constant 0 to produce an output 1, and the remaining two +1 transforms are not applied. Similarly,

when $X_1X_2 = 21$, then the middle +1 transform is applied, and when $X_1X_2 = 22$, then the right-most +1 transform is applied. The cost of this realization of carry output *C* is (5 + 1 × 2) + (5 + 1 × 2) + 5 = 19, and it requires one ancilla bit (the input constant 0 along the carry output line is not considered as an ancilla bit), which is not explicitly shown in Figure 2.30.

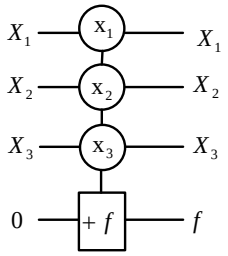
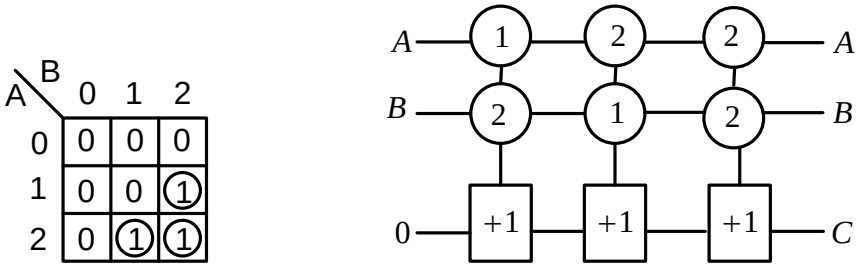


Figure 2.29. Realization of a single minterm of three ternary variables. Letters x_1, x_2 and x_3 denote the controlling values of variables X_1, X_2 and X_3, respectively.

Table 2.5 Ternary Truth table of a half-adder

AB	*CS*
00	00
01	01
02	02
10	01
11	02
12	10
20	02
21	10
22	11



(a) (b)

Figure 2.30 Realization of carry output C of ternary half-adder. (a) the ternary map of C, (b) its realization with double-controlled ternary Toffoli gates.

Until now, I have presented state-of-the art binary and ternary quantum circuits, with more emphasis on ternary gates. So far, only the above presented gates have been used. The dissertation now introduces more gates and synthesis methods using them.

2.6.4. Why and when can multiple-valued quantum circuits be used in quantum computing?

Why not multiple-valued quantum algorithms, their respective oracles, general blocks, circuits and gates? Although multiple-valued circuits are not very competitive in existing technologies, in the case of quantum technology, there is basically no limit to designing multiple-valued circuits. This is because they use the same physical phenomena as binary quantum circuits. I believe that more research in this direction has now become necessary. Some quantum realization technologies also allow the design circuits that will freely mix various radices and thus binary, ternary, and quaternary gates can co-exist in the same circuit. This leads to the concept of hybrid circuits. An obvious basic question thus arises – "how should future multiple-valued quantum circuits be designed?" There is already a small body of literature on multiple-valued quantum logic gates and

universality [DeVos02, Nielsen02, Perkowski02, Perkowski04], and synthesis algorithms [Brylinski01, Bullock05, Curtis04, Denler04a, Khan05, Khan04, Khan04a, Khan05c, Miller04, Miller04a, Perkowski02, Yen05].

The logic synthesis algorithms for multiple-valued quantum circuits can be divided into two groups**:**

(1) Designing the so-called permutative circuits (described by unitary matrices that are in addition permutative) [Curtis04, Denler04a, Khan04, Miller04, Miller04a, Perkowski02, Yen05],

(2) Designing circuits specified by arbitrary unitary matrices [Brylinski01, Bullock05, Khan05, Khan04a].

Although the first problem is a subset of the second one, it is more important in practice since all oracles in algorithms so far are permutative, and the non-permutative quantum blocks are usually standardized (like Hadamard or Fourier transforms), or do not require sophisticated design algorithms. The algorithms designed for the first group above use evolutionary programming principles or are adaptations of some methods used in Galois Field classical circuits and classical multiple-valued reversible logic [Curtis04, Denler04a, Khan04, Khan05c, Miller04, Miller04a, Perkowski02, Yen05]. The algorithms designed for the second group use evolutionary approaches or adapt the unitary matrix decomposition methods [Bullock05, Khan05] (matrix decomposition is the main approach for synthesizing general-purpose binary quantum circuits). All these methods are so far applicable only to quite small functions, rarely more than 5 – 6 qudits. There is also nothing in the literature on synthesizing certain types of practical special circuits, like arithmetic ones, using such quantum CAD tools or by hand. In contrast to

binary quantum design [Bruce02, Svore04, Vedral96], there are also no published examples of such circuits, with one exception [Miller04, Miller04a]. Similarly as in the binary quantum logic and classical design, researchers should therefore consider developing by hand examples of the most useful typical blocks; to aid future automated methods for some special categories of circuits and to evaluate the quality of the general-purpose MV logic synthesis algorithms. In classical computing much of the computer hardware is build from standard iterative blocks, and automated synthesis methods are used mostly for control circuits; at least that was the practice for many years. The PSU group believes that quantum computer design will be similar, and therefore I work on designing various "data path" building blocks of future quantum computers [Khan05b, Sazzad08] in my dissertation.

2.7. Background on hybrid binary-ternary quantum circuits.

In the previous sections I provided insight on binary and ternary quantum logic circuits. The concept of ternary logic can be generalized as a multiple value quantum logic circuit. In this section I present mixed quantum circuits. In the case of mixed binary-ternary quantum circuits, some wires can be presented as binary logic and some can be presented as ternary logic wires. Table 2.6 presents a truth table for a binary-ternary mixed quantum gate where x_1 is a binary logic wire while x_2, x_3 are ternary logic wires. The symbolic representation of the mixed gate is shown in Figure 2.31. When the value of qubit x_1 is 1 and the value of qutrit x_2 is 2, the *Z* transform is performed on wire x_3 (any qudit); the value is copied otherwise.

Table 2.6 Truth table of a mixed binary-ternary gate. The qudit x_1 is binary, qudits x_2 and x_3 are ternary.

$x_1x_2x_3$	$y_1y_2y_3$
000	000
001	001
002	002
010	010
011	011
012	012
020	020
021	021
022	022
100	100
101	101
102	102
110	110
111	111
112	112
120	$12\,Z\ of\ (x_3)$
121	$12\,Z\ of\ (x_3)$
122	$12\,Z\ of\ (x_3)$

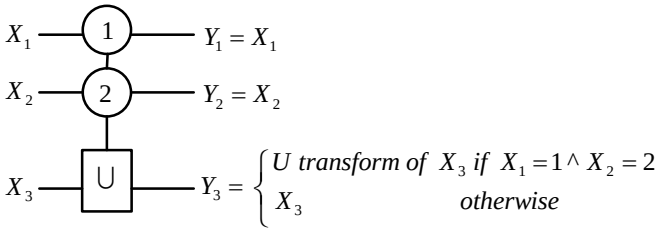


Figure 2.31. Realization of mixed binary-ternary gate using (a) symbolic double-controlled gate with binary qubit X_1 and ternary qutrit X_2.

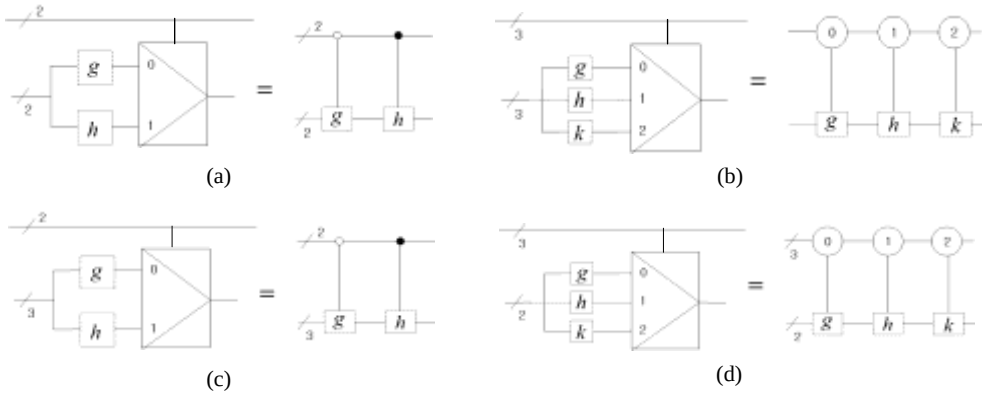


Figure 2.32. Mixed (hybrid) quantum muxes. (a) binary control and binary data and its realization from standard binary controlled quantum gates, (b) ternary control and ternary data and its realization from standard M-S gates controlled with values 0, 1, 2 respectively, (c) binary control and ternary data and its realization with M-S gates , (d) ternary control and binary data and its realization with M-S gates.

2.8. Conclusions and related work in the thesis.

Before I discuss the control functions and data functions for ternary logic, let us present the concept of the quantum multiplexer, called also a generalized gate or simple "Perkowski gate". Figure 2.32a presents the schematic of the binary quantum multiplexer. As we see it is equivalent to a sequence of two controlled gates. Figure 2.32b presents the ternary multiplexer and its corresponding circuit with controlled ternary gates. The analogy to the binary case is perfect. The values of ternary controls are shown inside the circles in the top wire. As we see, in the case of the binary mux the

controls are binary and the data are binary. In case of the ternary mux both controls and data are ternary. Similarly the technology allows the creation of mixed circuits with binary control and ternary data paths (Figure 2.32c). I call these binary-control/ternary-data or 2/3 circuits. Figure 2.32d presents a ternary control-binary-data quantum mux. All our circuits are based on these constructions. In the case of ternary logic the controls c_i are the controls of gates as shown in Figure 2.32a and the data are the data from circuits like from Figure 2.32a.

From the point of view of handling information, ternary quantum technology is theoretically the most desirable choice for future computing systems. However, similarly as in CMOS technology, an argument for quaternary logic is the ease of binary-quaternary conversion [Perkowski04]. Also, the nature of entanglement is different in binary, ternary and quaternary logics. Finally, there are physical phenomena where very large radices may be realizable for qudits [Muthurkrishnan00]. It is therefore our opinion that because it is very early to speculate how practical multiple-valued quantum circuits will be built, the theory should investigate all possible approaches and find theoretical and experimental ways of comparing them on large examples of practical usefulness. So far, only small artificially or randomly created MV functions have been used as benchmarks for synthesis. For the purpose of this dissertation, I have chosen Ion Trap realizable [Klimov03, Muthurkrishnan00] 1-qutrit ternary permutative gates as the elementary building blocks, which change the state of a single qutrit. I have also chosen a family of Ion Trap realizable ternary 2-qutrit controlled Muthukrishnan-Stroud gates [Muthurkrishnan00] as the elementary building blocks. These gates apply a 1-qutrit

permutative transform on the controlled qutrit when the controlling qutrit is 2. The physical realization of these gates is best presented in the literature, as compared to other potential multiple-valued technologies. The solutions presented in the next chapters create ancilla qudits in more than minimal numbers, so they decrease the number of gates at the cost of adding more qudits to our circuits. Normally in the literature one tries to minimize the number of ancilla bits, but it is not known how important the number of these bits will really be for future solid state Ion Trap quantum technologies [Monroe95, Wineland98, Leibfried03]. The circuit with more ancilla bits has a smaller number of gates, so again I believe that having no ancilla qudits should be not a dogma in quantum circuit synthesis. I believe that various trade-offs of the number of gates, number of ancilla bits and number of measurements should be investigated on practical circuits of non-trivial size. Techniques such as constant overlap and local mirrors should be investigated in these comparisons [Shivgand05].

Multiple-valued Quantum circuits are a promising choice for future quantum computing technology since they have several advantages over binary quantum circuits. Quantum algorithms can be constructed more efficiently in ternary quantum than in binary quantum logic, including the Deutsch-Jozsa algorithm or the practically useful Grover algorithm. This requires an ability to design oracles from ternary quantum circuits. There therefore exists a need to develop software to simulate, design and test ternary and in general multiple-valued quantum circuits. The goal of the entire research direction started in this dissertation is to create improved CAD tools for binary, ternary and quaternary quantum circuit synthesis. These tools should use ternary Feynman and Toffoli gates and

other gates built on top of the ion trap realizable ternary 1-qudit and Muthukrishnan-Stroud gates. However, only binary software has been developed and tested. I have obtained very good results. The general software for other values can be written on similar principles (as our algebraic methods are unified) and is left for future work.

CHAPTER 3

Binary Tree and Decision Diagram Expansions for Quantum Circuits Synthesis.

It is well-known [Sasao93e, Sasao95f, PHDSazzad] that Reed-Muller logic is fundamental in synthesizing binary reversible and permutative quantum circuits. Its study may be useful for improving synthesis algorithms and designing new algorithms for multiple-valued quantum circuits. A number of new algebraic families of complete operator sets and circuit structures, based on the Reed-Muller and Galois Field type logics, with particular interest in multi-valued logic ESOP-like expressions (non-canonical) and their (canonical) subsets, as well as tree expansions, are introduced below in Section 3.1 as the starting point for the explanation of the original synthesis methods to be introduced later.

3.1. Types of Logic.

Several different algebraic systems of group based logics, from the well-known Boolean to the Galois field types, are presented in this section, based on the literature. Group based logic is the logic in which the combining operator is a group in the sense of abstract algebra (see definition below). Classical Boolean Logic, traditionally utilized in classical logic circuit design, is restricted to binary input/output values and a few simple, basic gates. Galois Logics (non-quantum) can be viewed as a generalization of Boolean Logic, as Galois Field mathematical operations are applicable for multi-valued logic, over an arbitrary finite field. Also, as discrete, multi-valued numbers (nominal values) are used in Galois Field mathematics, in the quantum world it provides a more natural circuit implementation than that of Boolean logic.

Even in classical CMOS design, the AND-EXOR implementation has been shown to be more economical than AND-OR, generally requiring fewer gates and connections. This logic is also highly testable, adding powerful arguments to the case for being a particularly good candidate for quantum circuit design. It is for these reasons that the research done in this dissertation will generate AND-EXOR-like structures for multi-valued quantum logic. First, the binary structures will be presented for completeness and as an introduction to our methods for both binary and MV.

It is well known that the AND-EXOR form has been developed into a complete hierarchy of Reed-Muller (RM) Expansions, using the Shannon [Shannon49], Positive Davio and Negative Davio Expansions. This hierarchy is described with logic equation forms, trees, and decision diagrams [Sasao93e]. The AND-EXOR representations have interesting characteristics, allowing the inexpensive construction of large functions and efficient representation of their properties.

To incorporate the multi-valued logic method, the Reed-Muller form was extended for Galois Fields [Stankovic97]. Galois theory demonstrates that for every number k^n, where k is prime and $n \geq 2$, there exists a unique matrix under mathematical operators, such that all group theory properties are satisfied. It is then apparent that the technique that can be utilized for the representation of binary logic can also be used for the multi-valued logic. The research specifying the relationship between the Reed-Muller Hierarchy and Galois Fields [Dill97, Dill97a, Edward93, Kalay98, Kalay98c, Khan03, Khan04] provides the capability of expressing multi-valued logic with decision diagrams and compact algebraic

structures (or canonical forms), which can be represented with "universal" expansion circuits. Such circuits can be mapped to structural binary technology [Tsai96]. There exits a Galois Field Sum-of-Product form that is analogous to the binary ESOP. The GF Sum-of-Products is the most general two-level form for expressing a Galois Field multi-valued logic circuit. This form is both highly testable and allows the most efficient two-level implementation. From this form, a logical hierarchy can be developed and represented with expressions, trees, decision diagrams, and two-level forms. Hence a Generalized Galois Hierarchy is obtained as the result of extending the Generalized Reed-Muller concepts to arbitrary Galois Fields.

Reed-Muller Logic Theory was also expanded with the introduction of Generalized Kronecker Expansions to the Zhegalkin Hierarchy [Perkowski97a, Perkowski97c] (Note that the Zhegalkin Kronecker Reed-Muller Form is obtained when a single expansion, from the set of all possible Zhegalkin (Linearly Independent (LI), AND-EXOR canonical form) expansions, is applied in every level.). It was also recognized that *"the GRM expansion with functional coefficients is a special case of the LI expansion with functional coefficients"* [Perkowski97b]. The Zhegalkin Hierarchy is a subset of the Linearly Independent Logic Hierarchy [Perkowski97b]. It includes the Reed-Muller Hierarchy and all other (known and future) AND-EXOR forms. (Note that the Linearly Independent Hierarchy is not restricted to circuits built from AND and EXOR gates in the binary case.) The Zhegalkin Hierarchy is named in honor of the Russian scientist who in 1927 discovered the forms [Zhegalkin29] now attributed to Reed and Muller's research published in 1954 [Reed54].

At the present time, multi-valued hardware realizations are not commonly available. There is some research on commercial multi-valued hardware circuits such as memories, FPGAs, and arithmetic operation chips that are emerging in the marketplace [Beers98, Current95, Hasuo92, James87, Michael92, Zilic93]. This author remains skeptical however, about the future of multi-valued logic synthesis for hardware design in classical technologies, while the quantum technology is very promising for multiple-valued logic, as discussed in the previous and next chapters. It is thus important not only to have a strong new logic theory, but also to have a methodology for efficient synthesis and minimization of circuits in a logic which is highly testable. Because of high integration, those technologies that are highly testable, fault tolerant and repairable will win the future competition for quantum hardware and will become most prevalent.

Currently, logic synthesis and minimization applications in standard binary hardware such as ASICs and FPGAs, can be implemented for Galois Logic only in fields of $GF(2^n)$. A practical method for implementing Galois Logic in GF(3), GF(4) and GF(8) is shown in this thesis. For all mathematical operations of Galois Logic in GF(3) and GF(4), the quantum implementation is demonstrated to require relatively few gates. Since the binary implementation of all the operations in the new Galois Logic are simple and compact, they are easily realizable in quantum gate-based regular layouts, with minimal interconnections. Further, as multi-valued quantum hardware becomes readily available, it is expected that these same good design characteristics will be realizable for the multi-valued logic implementation with high radices.

The Ring Algebra and Min Modsum circuits use the same modulo addition operation but a different multiplication operation. They both allow an implementation of multi-valued logic with radices 3, 6 and 8. Ring Algebra consists of two mathematical operations; ring addition and ring multiplication. MinModsum uses minimum (MIN) as multiplication. Together they constitute two families of simple gates, creating two complete logic systems. Both modulo addition (Modsum) and GF addition are group operators. Linearly independent functions can be constructed within this system, providing a means of creating universal modules and the corresponding expansions. The Ring Sum-of-Product form, for both generalized and universal literals, is known for its compact form and good testability properties [Kalay99b].

In this research, the Galois and controlled gate, multi-valued logic families will be presented. I build all families of controlled gates but do not use them on higher level of synthesis, as I do not know how to use controlled gates in expansions that are the base of my approach. The invention process for these new algebras will also be fully explained. This should allow the careful reader to apply the same methods in case more new algebraic structures, similar to fields, are invented in future; they will also be realizable in any quantum technology. It will be demonstrated in a systematic way that given the algebra, with mathematical operations constituting a complete logic system, providing the capability to construct linearly independent functions, there are many ways to create universal gates and the corresponding expansions. AND-EXOR logic structures are utilized for their good match to quantum technology. With AND-EXOR-like logics, the

Sum-of-Products-type forms for the given algebras are the most general, regular (for possible hardware layout purposes), and highly testable forms for expressing a given function. Thus, the particular algebra's SOP (like "modulo-sum" of "min" products) is a good form to develop the logic family hierarchy. Methodically, starting from expansions, then trees, followed by decision diagrams, and finally two-level forms, the complete new linearly independent, multi-valued logical family hierarchies will be developed. The relations between the Galois Field and Controlled Gate families are shown in Figure 3.1. This method of logic family invention may also be applied for other group-based logics.

Definition 3.1.1. A group (G,*) is a non-empty set G and a binary operator (*) on G, such that the following conditions hold:

Closure: For all a, b $\in$ G, a*b $\in$ G.

Associativity: For all a, b, and c in G, a * (b * c) = (a*b) * c.

Identity: There exists an identity element "e" $\in$ G, such that a * e = e * a = a, for all a $\in$ G.

Inverse: For each a $\in$ G, there exists an inverse $a^{-1} \in$ G, such that a * a^{-1} = a^{-1} * a = e.

More on groups can be found in [Stewarts89].

Definition 3.1.2: A non-empty set F is called a field when the two binary operations "+" and "*" are defined on F and the following properties hold.

Closure: For all a, b $\in$ F, a + b $\in$ F and a*b $\in$ F.

Associative: For all a, b, and c $\in$ F, a + (b + c) = (a + b) + c and a * (b * c) = (a * b) * c.

Commutative: For all a, b $\in$ F, a + b = b + a and a * b = b * a.

Distributive: For all a, b, and c $\in$ F, a * (b + c) = a * b + a * c and (a + b) * c = a * c + b * c.

Identity: For all a $\in$ F, a + 0 = a, (since 0 is the additive identity) and a * 1 = a, (since 1 is the multiplicative identity).

Inverse: For each a $\in$ F, there exist inverses –a $\in$ F and $a^{-1} \in$ F, such that a + (-a) = 0, (since –a is the additive inverse) and a * a^{-1} = 1, (since a^{-1} is the multiplicative inverse). A finite field is also called a Galois Field. It can only have r^k elements, where r is a prime and $k \geq 1$ is a natural number. In this thesis we are interested only in finite fields, but other researchers work on infinite quantum fields as well.

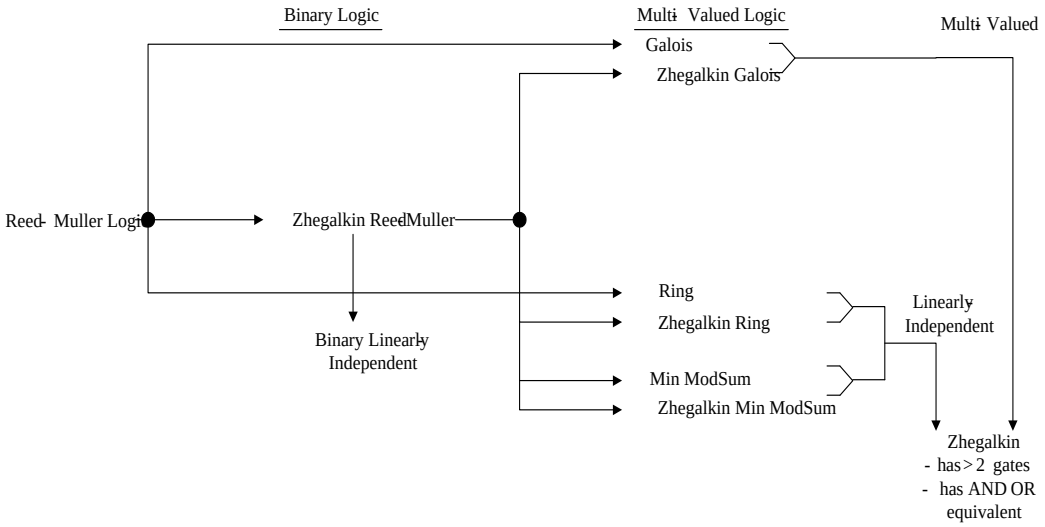


Figure 3.1 Algebras and logic families that are of interest to this dissertation. At this stage the research on Min-Modsum and Rings is less developed but we believe that similar properties will be found in other families.

3.2. Binary Reed-Muller Logic.

The Reed-Muller Hierarchy is well-known from the literature [Sasao95f]. Therefore only a limited review of the definitions for the hierarchy of forms will be included. For classical digital logic design, the AND-EXOR implementation has been shown to be

economical, generally requiring fewer gates and connections than that of AND-OR logic. The gate structure also provides for ease of testability, making it desirable for classical FPGA designs. There are several different classifications of AND-EXOR expressions. All Reed-Muller forms are canonical, as only one functional expression exists for the polarity of the specified variables. The general expression for all of the Reed-Muller forms is given as follows:

$f(x_1,x_2, \ldots, x_n) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_nx_n \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus \ldots \oplus a_{n-1,n}x_{n-1}x_n \oplus a_{12\ldots n}x_1x_2x_3\ldots x_n$

where the "a" coefficients are binary constants.

Function expansions are the basis for the derivation of the hierarchical, logic family. The Shannon, Positive Davio, and Negative Davio are well known. Given an arbitrary logic function, $f\ (x_1, x_2, \ldots x_n)$, where $f_0 = f\ (0, x_1, x_2, \ldots x_n)$, and $f_1 = f\ (1, x_1, x_2, \ldots x_n)$, and $f_2 = f_0\ (x_1, x_2, \ldots x_n) \oplus f_1\ (x_1, x_2, \ldots x_n)$, these expansions are defined as follows.

Positive Davio: $f = f_0\ (x_1, x_2, \ldots x_n) \oplus x_1 [\ f_0\ (x_1, x_2, \ldots x_n) \oplus f_1\ (x_1, x_2, \ldots x_n)\ ]$

(Equation 3.1)

Negative Davio: $f = f_0\ (x_1, x_2, \ldots x_n) \oplus \overline{x_1} [\ f_0\ (x_1, x_2, \ldots x_n) \oplus f_1\ (x_1, x_2, \ldots x_n)\ ]$

(Equation 3.2)

Shannon Expansion: $f = \overline{x_1}\ f_0\ (x_1, x_2, \ldots x_n) \oplus x_1 [\ f_0\ (x_1, x_2, \ldots x_n) \oplus f_1\ (x_1, x_2, \ldots x_n)\ ]$

(Equation 3.3)

The following definitions describe each of the specific forms within the Reed-Muller Hierarchy. The most restrictive classification is the Positive Polarity Reed-Muller

(PPRM) expression form. This type of equation is canonical, with no more than 2^n products of positive literals. It is more formally defined as follows:

Definition 3.2.1: The Positive Polarity Reed-Muller Form (PPRM) is a Reed-Muller expression in which all literals of variables are expressed with positive polarities. The example of the expansion tree using only Positive Davio Expansions is given in Figure 3.2.

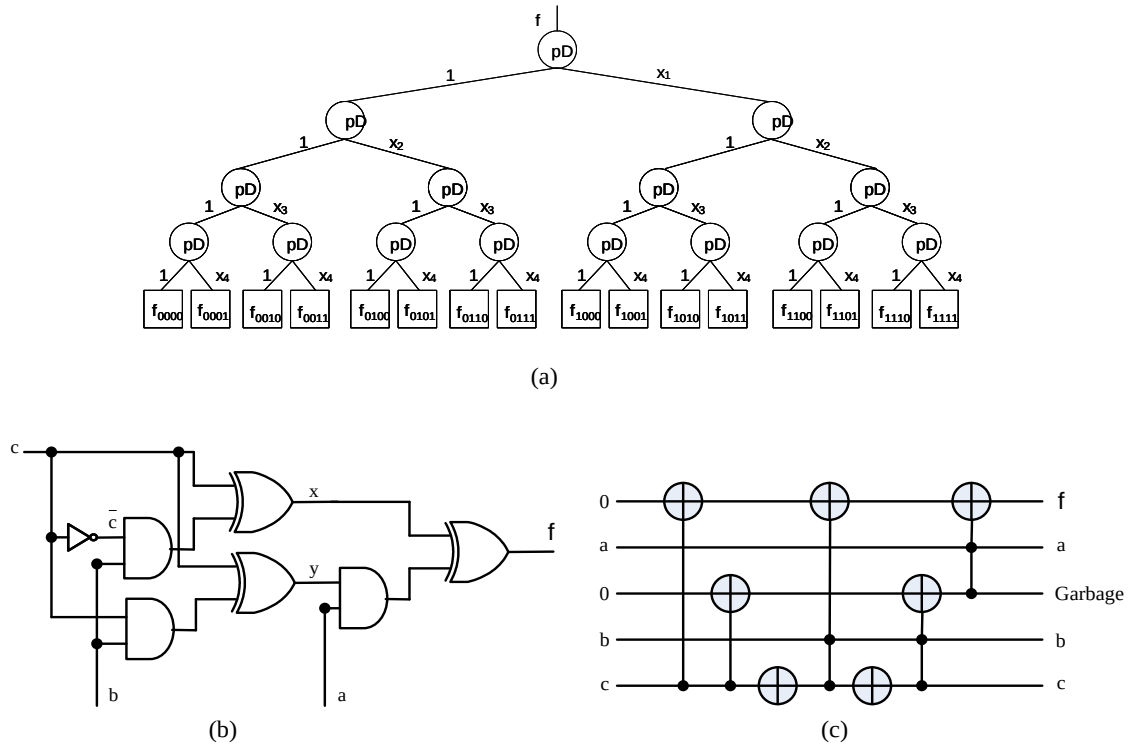


Figure 3.2. (a) Expansion tree with Positive Davio Expansion only, (b) An example of diagram with positive Davio gates (c) A quantum array directly corresponding to the classical diagram from Figure 3.2 (b). All Toffoli gates are 3x3 Toffoli gates only. One ancilla bit is used.

Example 3.1: An example PPRM function $f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_4 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_3 \oplus x_2x_3x_4$ is given in Figure 3.3. Figure 3.3 (a) presents the AND-XOR circuit implementation of the function, Figure 3.3b presents

the pD (Positive Davio) decision diagram for the function and Figure 3.3c illustrates a quantum array created from the decision diagram.

Inclusive of the PPRM types of expressions, another classification of AND-EXOR equations is the Fixed Polarity Reed-Muller (FPRM) form. The FPRM form (expression) can contain either positive or negative literals for each variable, but not a mixture of terms with various polarities of the same variable. The expression is canonical, having unique coefficients, and usually requires fewer terms than the PPRM form, but can never have more.

Definition 3.2.2: The Fixed Polarity Reed-Muller Form (FPRM) is defined as a Reed-Muller expression in which all the literals of a variable can be either positive or negative, but cannot exist in both polarities.

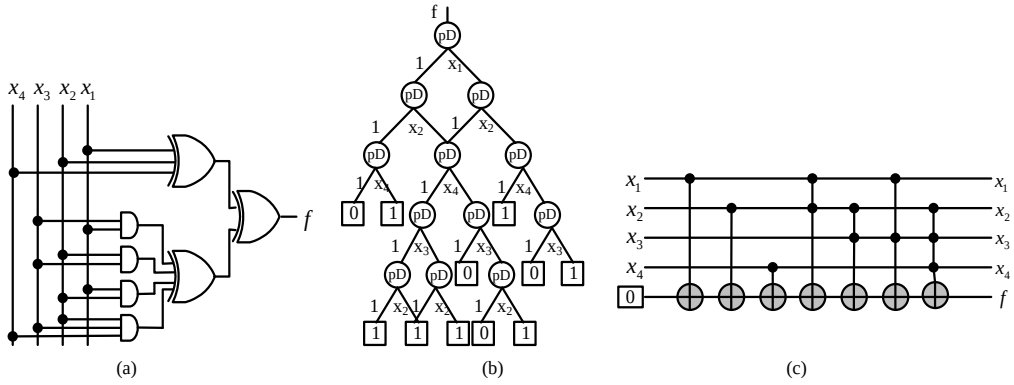


Figure 3.3(a) AND-XOR circuit implementation of PPRM function in Example 3.3.1 (b) Positive Davio Decision Diagram presentation of the function (c) Quantum Array realization of the expressions.

Example 3.2: An example of a FPRM is given.

$$f(x_1, x_2, x_3, x_4) = \bar{x}_2 \oplus x_3 \oplus x_3\bar{x}_4 \oplus x_1x_3 \oplus x_1\bar{x}_4$$

The variables x_2 and x_4 have negative polarities, while x_1 and x_3 have a positive polarity throughout the expression. Figure 3.4 presents an FPRM Davio decision diagram, circuit and quantum array respectively.

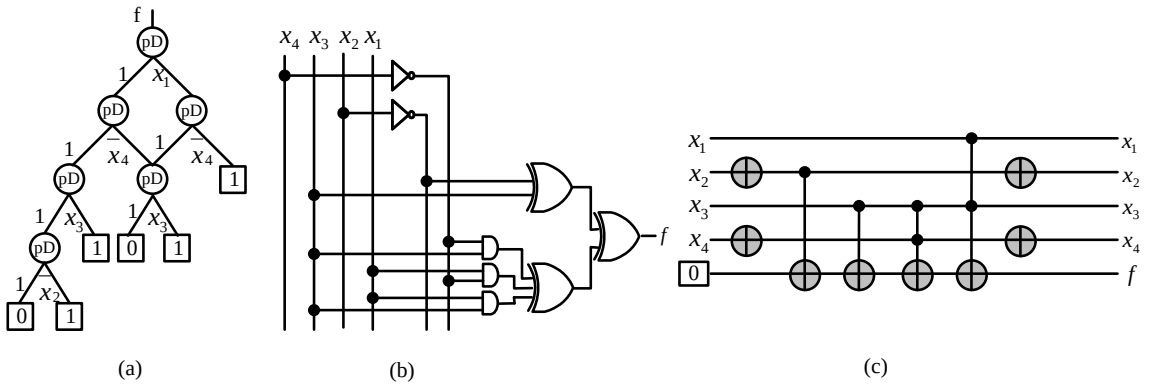


Figure 3.4 (a) Fixed polarity Davio Decision Diagram presentation of the function (b) AND-XOR circuit implementation of FPRM function in Example 3.2 (c) Quantum Array oracle realization of the FPRM obtained from the Decision Diagram from Figure 3.4a. As this is a quantum oracle, the input variable values are restored.

Given an arbitrary Reed-Muller Expression, to formulate the FPRM expression, either Positive or Negative Davio Expansions are applied for each variable x_i (i = 1, 2, 3, ..., n).

Example 3.3: The FPRM form is derived for the expression given.

$$f(a,b,c,d) = \bar{a}b\bar{c}d \oplus a\bar{b}c\bar{d}$$

Substitutions for literals of variables, with identity expressions, are made as follows:

$$\bar{a} = a \oplus 1$$
$$\bar{b} = b \oplus 1$$
$$c = \bar{c} \oplus 1$$
$$d = \bar{d} \oplus 1$$

in order to obtain the FPRM circuit in polarity ($a, b, \bar{c}, \bar{d}$).

Thus,

$$f(a,b,c,d) = \bar{a}b\bar{c}d \oplus a\bar{b}c\bar{d}$$

$$f(a,b,c,d) = (a \oplus 1)b\bar{c}\,(\bar{d} \oplus 1) \oplus a(b \oplus 1)(\bar{c} \oplus 1)\bar{d}$$

$$f(a,b,c,d) = (ab \oplus b)(\bar{d}\,\bar{c} \oplus \bar{c}\,) \oplus (ab \oplus a)(\bar{c}\,\bar{d} \oplus \bar{d})$$

$$f(a,b,c,d) = ab\bar{c}\,\bar{d} \oplus b\bar{c}\,\bar{d} \oplus ab\bar{c} \oplus b\bar{c} \oplus ab\bar{c}\,\bar{d} \oplus a\bar{c}\,\bar{d} \oplus ab\bar{d} \oplus a\bar{d}$$

$$f(a,b,c,d) = b\bar{c}\,\bar{d} \oplus ab\bar{c} \oplus b\bar{c} \oplus a\bar{c}\,\bar{d} \oplus ab\bar{d} \oplus a\bar{d}$$

Definition 3.2.3: The Kronecker Reed-Muller Form (KRO) is derived by the application of Shannon, Positive Davio, or Negative Davio Expansions, with the restriction that only one type of expansion can be applied per level in an ordered tree. An example of a KRO expansion tree is shown in Figure 3.5.

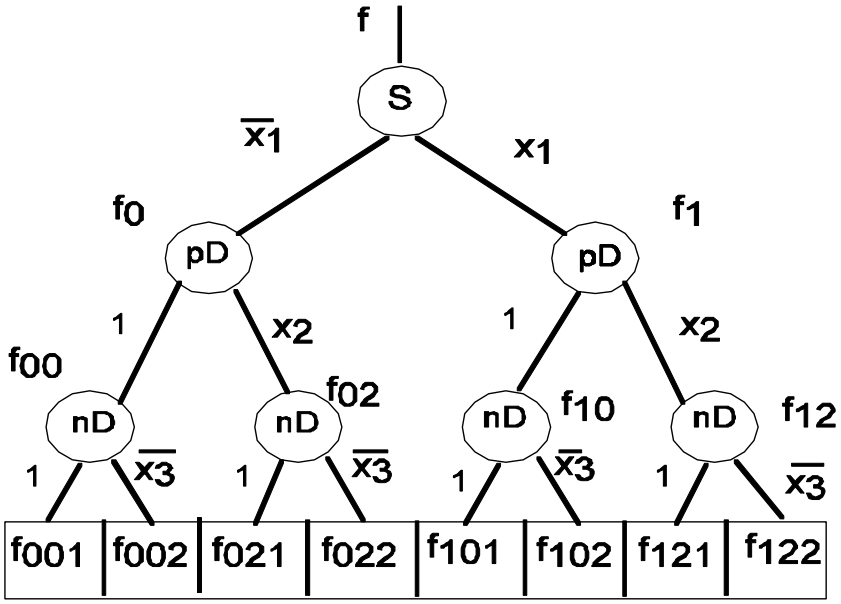


Figure 3.5 Example of a KRO Expansion tree.

An algebraic expression can be constructed from an expansion tree by combining the coefficients for each expansion, with multiplication, along each branch to the leaf and then combining the branches with the exclusive-or operation.

Example 3.4: Write the algebraic expression for the example KRO Expansion tree shown in Figure 3.5.

$$f(x_1,x_2,x_3) = \overline{x_1}\cdot 1\cdot 1\cdot f_{001} \oplus \overline{x_1}\cdot 1\cdot \overline{x_3}\cdot f_{002} \oplus \overline{x_1}\cdot x_2\cdot 1\cdot f_{021} \oplus \overline{x_1}\cdot x_2\cdot \overline{x_3} f_{022} \oplus x_1\cdot 1\cdot 1\cdot f_{101} \oplus x_1\cdot 1\cdot \overline{x_3} f_{102} \oplus x_1\cdot x_2\cdot 1\cdot f_{121} \oplus x_1\cdot x_2\cdot \overline{x_3}\cdot f_{122}$$

Definition 3.2.4: The Pseudo Reed-Muller Form (PSDRM) is obtained by applying the Positive Davio and Negative Davio Expansions, but with different expansions for each sub-function, as desired. This means that observing the expansion tree for the PSDRM, either Positive or Negative Davio expansions are applied at will, regardless of the node or level of the tree, and different types of expansions may be used for the same variable. An example of a PSDRM Expansion Tree is given in Figure 3.6. In this tree, each level consists of both Positive and Negative Davio Expansions.

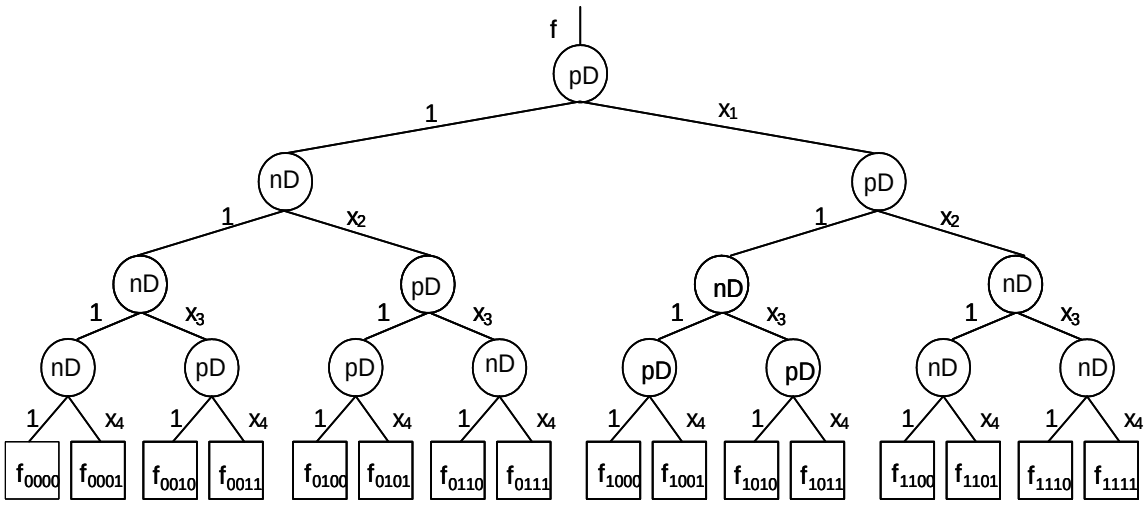


Figure 3.6 Example of a PSDRM tree.

Example 3.5: For the function $f(x_1,x_2,x_3,x_4) = x_1\ x_2\ x_3\ x_4 \oplus \overline{x_1}\ \overline{x_2}\ \overline{x_3}\ \overline{x_4}$, the particular PSDRM form described by the tree in Figure 3.6 can be applied to produce the expansion tree shown in Figure 3.7.below. The classical circuit for $x_1\ x_2\ x_3\ x_4 \oplus \overline{x_1}\ \overline{x_2}\ \overline{x_3}\ \overline{x_4}$ is given

in Figure 3.8 and the quantum array for the flattened PSDRM form is shown in Figure 3.9.

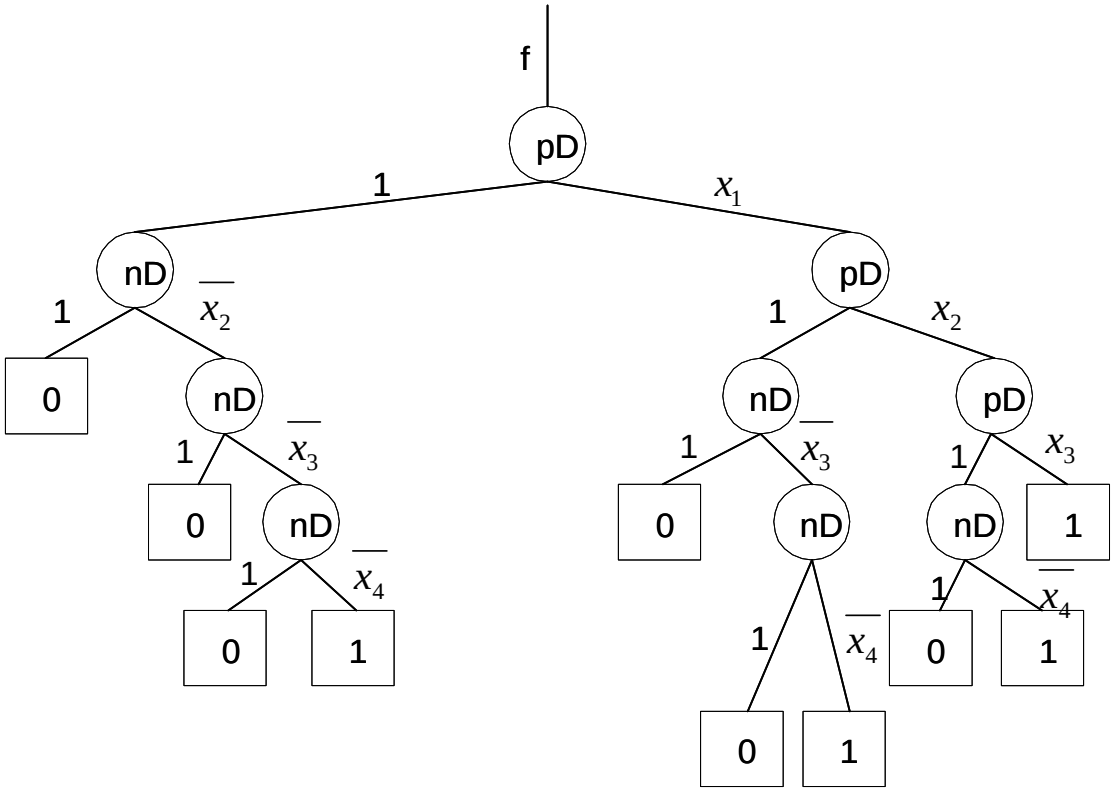


Figure 3.7. $f(x_1, x_2, x_3, x_4) = x_1\ x_2\ x_3\ x_4 \oplus \overline{x_1}\ \overline{x_2}\ \overline{x_3}\ \overline{x_4}$ *in certain (one of many) PSDRM form. The flattened form* $f(x_1, x_2, x_3, x_4) = x_1\ x_2\ x_3\ x_4 \oplus \overline{x_1}\ \overline{x_2}\ \overline{x_3}\ \overline{x_4}$ *is illustrated in Figure 3.7b.*

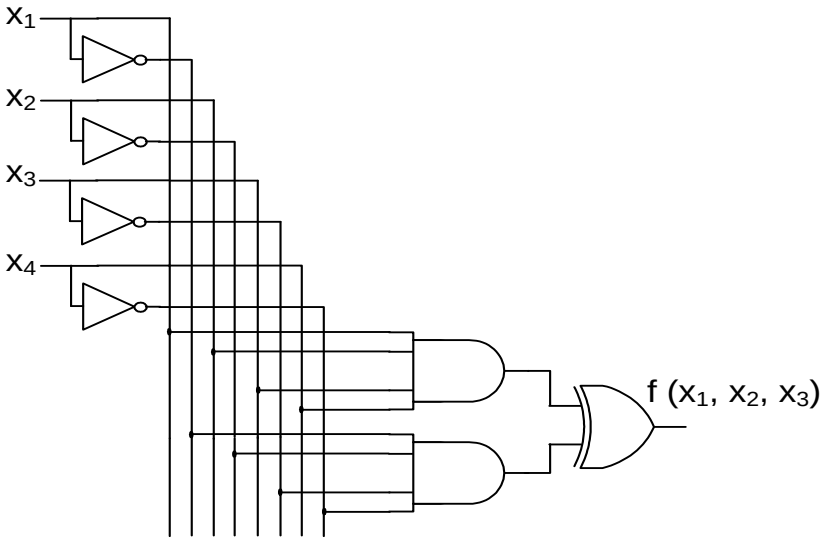


Figure 3.8 AND-XOR circuit implementation of function from Example 3.5.

Definition 3.2.5: The Pseudo-Kronecker Reed-Muller Form (PSDKRO) is derived by the application of any subset of Shannon, Positive Davio, and Negative Davio Expansions in an expansion tree level, but the tree is ordered (with the same order of variables for each branch). Herein, the tree is defined and introduced as desired.

Definition 3.2.6: The Free Kronecker Reed-Muller Form (FKRM) is not canonical and is derived by the application of the Shannon, Positive Davio, and Negative Davio Expansions, without restrictions and with no ordering of variables.

Concluding this chapter, we presented here, based on the literature, various concepts used in the design of binary AND/EXOR circuits that have already been extended to quantum circuits in previous research of the PSU group. In this dissertation I use all these concepts and I generalize them to multiple-valued logic quantum circuits. In this chapter I presented also my own discussion of previous synthesis ideas and what can be done in my opinion as new research for quantum circuits. These new possible developments are, however, not the main core of my dissertation work. The next chapters will show the main work of this dissertation.

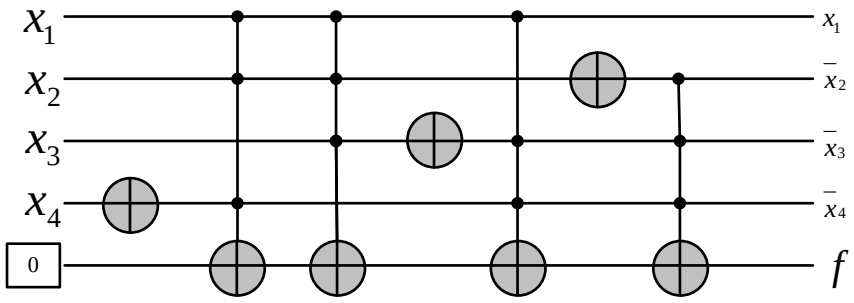


Figure 3.9. A quantum array being not an oracle for flattened PSDRM from Figure 3.7.

CHAPTER 4

Towards Efficient Computer Aided Design Methods for D-level Quantum Computers.

4.1. Towards Computer Aided Design of d-level Quantum Computers.

In chapter 2 we introduced the Ion Trap and d-level quantum computing technologies in the full detail necessary for the research of this dissertation, and especially for the development of quantum synthesis CAD tools. This was done based on the recent literature and in this aspect I claimed no innovation – I just reported on all the technical details that will be used in this thesis to create new circuits, synthesis algorithms and synthesis methods for quantum circuits. Here I will give some new answers to the synthesis questions, and these answers will lead to the next chapters of this dissertation.

In chapters 2 and 3, I introduced binary, ternary and hybrid quantum logic gates and I build several simple circuits using these gates. I explained the minimum mathematical foundations for building these quantum circuits and showed some simple design principles. Finally, I discussed what influence the particular technology for quantum realization such as Ion Trap has on the logic synthesis of such circuits. All synthesis methods need some kind of search. Obviously, everybody familiar with classical search algorithms such as A* search, Depth-first search, Breadth-first search, deepening search, Genetic Algorithm, Particle Swarm Optimization (PSO), Bacteria Foraging (BF) or other search algorithms such as Simulated Annealing, Tabu Search or Scattered Search can create algorithms of these types that would be more efficient than the exhaustive algorithm mentioned in chapter 3. An important aspect was thus missing from chapters 2

and 3: I did not show any efficient synthesis algorithm and the mentioned search algorithm was applicable only to relatively small circuits.

In future, with the capability to fabricate larger and larger quantum circuit designs, there will be an increased need for high testability, fault-tolerance and self-repair. Increasing design complexity and more aggressive time-to-market schedules will place higher demands on the human design teams that design the quantum computers. Therefore, it is very likely that the design methodology for quantum computers will be quite similar to the design methodology for 2010 standard VLSI computers. Thus, new methods of design become necessary, which will combine both human expertise with that of the increased computational capabilities of modern (classical) computers. Computer Aided Design methods should be developed for quantum computers and in this dissertation I relate them to the particular breakthrough technology of Ion Trap quantum computers.

4.2. How to synthesize quantum circuits on level of pulses, simple gates and circuit blocks?

In Chapter 2 we explained about the low level synthesis of quantum circuits. Two important questions are:

1. how to specify such circuits in a way that is convenient to the quantum computer designer at the level of the device, circuit or quantum array block,
2. how to convert a (higher level, more abstract) specification into an optimized circuit with gates, and, ultimately, into the low level physical and geometrical descriptions required by quantum Ion Trap technology.

Various methods have been proposed in general for quantum and reversible circuits, of which the historically first and so far the most advanced are the methods of evolutionary algorithms, specifically Genetic Algorithms and Genetic Programming. There are, however, no papers on designing binary and d-level quantum circuits specifically for the constraints and advantages of Ion Trap technology. This dissertation attempts to fill this gap.

It is well known that Genetic Algorithm (GA) [Goldberg89, Holland92] and Genetic Programming (GP) [Koza92, Koza94, Koza99] techniques provide a means for applying the theory of evolution within an artificial system. GA is a system that evolves problem parameters directly; GP evolves programs for problem solution. Through a process of emergent intelligence, GA/GP formulates engineering solutions based on an accumulated knowledge of the problem and the merit of potential solutions. In recent years Genetic Algorithm and Genetic Program have been successfully applied to a wide range of engineering problems and were the main methods used in other research groups working on (binary) quantum circuits design [Rubinstein01, Spector99, Williams99]. They were also used in our research group for quantum circuit synthesis [Lukac05, Lukac05a, Lukac07, Lukac07a, Khan03, Khan04, Khan05, Khan05a, Giesecke07, PHDSazzad], including synthesis of multiple-valued quantum circuits for NMR technology [Giesecke07, Khan07, Khan07a] (NMR is similar in some respects to 1D layout of Ion Trap).

The other published papers about MV quantum synthesis came from Miller's and Bullock's teams and are not based on evolutionary paradigms but on certain group theory heuristics [Miller03] or on algebraic matrix decomposition theory [Miller04, Miller04a, Miller06, Bullock05]. Similarly the work of Faisal Khan at PSU uses the Sine-Cosine decomposition adapted from the discipline of numerical algebra. However, all these methods brought only limited success to the design of quantum circuits, as they do not use knowledge and engineering design ideas sufficiently. It was therefore the assumption and the starting point of this dissertation to avoid evolutionary algorithms and develop new algorithms based on combining knowledge-based and heuristic methods to achieve better results for larger circuits realized with the specific, Ion Trap, technology in mind. We do this, however, at the cost of adding more ancilla bits and being satisfied with approximate quality solutions rather than expecting exact solutions.

Past experience has shown that the application of pure evolutionary algorithms to logic minimization had serious limitations of size, computation time, and solution optimality [Dill97, Dill98]. Therefore, the PSU team also tried group theory methods and methods based on exhaustive search. These methods were successfully applied by our PSU research group for synthesis of quantum circuits, including multiple-valued circuits [Giesecke07]. So far, these methods have been applied only to small circuits. Yet another type of methods are the heuristic tree search methods such as the A* algorithm that is widely used in the area of classical (Good Old Fashioned) Artificial Intelligence [Lukac03]. Before we discuss our new synthesis algorithms we will systematically introduce the necessary background on logic gates, expansions, forms and structures.

4.3. Reed-Muller Logic, Galois Logic and Quantum Computing.

The fundamental gates in binary quantum circuits are:

- all single-qubit (rotation) gates (like Pauli rotations) including the NOT operator of Boolean logic,
- CNOT (Controlled NOT) which uses the EXOR operator of Boolean logic,
- Toffoli (which uses double-controlled NOT or C = ab ⊕ c function) and
- generalized Toffoli, like A=a, B=b, …, M = abcde…n ⊕ m.

These gates are internally build from Controlled-V (Controlled Square root of NOT) and its adjoint gate Controlled-V^+. These in turn are built from the 2-qubit interaction gate and single qubit rotations. Thus, the component gates that are really available in quantum hardware are: the single qubit rotations and the interaction gate. In one-dimensional (1D) "linear" Ion Trap technology the circuit must satisfy the "neighborhood constraint", i.e., all gates are only single-qubit or 2-qubit realized on neighbor qubits. This is a very serious design constraint, not taken into account by logic synthesis researchers so far. It is important to understand that every binary, ternary, quaternary, hybrid etc quantum circuit is built only from these primitives and including the neighborhood constraint. When we generalize these methods to MV logic, the only requirement of such circuits is that the higher the logic radix is, the more numeric values of discrete angles must be used in the single-qubit rotation gates. The pulse-level gates are the "quantum end" of synthesis. What is, however, the other end – the "abstract model" that should be used with this low-level quantum description? This model must be a good match with the linear Ion Trap technology that dominates now, but must be expandable to future 2D and 3D Ion Trap layouts. In this dissertation I want to find such models.

The algebra of EXOR and controlled circuits (with commutative operations like *(a ⊕ c)* and non-commutative operations like *(a CONTROL c)* is not very similar to the well-known AND/OR/NOT Boolean logic. Thus the classical AND/OR/NOT logic design/optimization methods such as finding prime implicants, graph coloring or unate/binate covering approaches are not directly applicable to quantum logic synthesis. The reason for this is that these methods are not linked to a useful representation algebra and are not linked to the underlying physics of quantum pulses and constraints of quantum layout. Therefore, new methods based on linear algebra, group theory and enumerative methods have been developed at PSU; they are the background of my research presented here. In contrast to classical CMOS logic where the realization of the EXOR operator is rather expensive, the gates based on EXOR are the cheapest gates in all quantum technologies, because of the similarity of operation of this gate to the interaction of two particles (recall examples from Chapter 2). As I will demonstrate in this dissertation, this property can also be generalized to multiple-valued and hybrid quantum circuits.

The Exclusive-Or Sum-of-Product (ESOP) expression is the most general, unrestricted AND-EXOR logic expression. It is not-canonical, in contrast to other forms presented here. ESOP compares favorably in classical digital design to AND-OR Sum-of-Product (SOP) logic, which is, however, more widely used. In the case of quantum circuits, AND-EXOR logic is of fundamental importance and the use of AND-OR logic is very limited. Let me cite:

Functions realized by such circuits (ESOPs) can have fewer gates, fewer connections, and take up less area in VLSI and especially, FPGA realizations. They are also easily testable [Fujiwara86, Pradhan87] It was shown, both theoretically and experimentally [Salmon89, Sarabi92, Sasao91, Sasao90, Sasao91a] that ESOPs have on average smaller numbers of terms for both "worst case" and "average" Boolean functions. It was also shown that ESOPs and all their sub-families have their counterparts in logic with multiple-valued inputs…

Additionally, as shown in [Kumar07], quantum circuits based on two-level AND-EXOR realizations are also good for the combinational logic portions of Finite State Machines, as they have proven more testable and can yield less area than do two-level AND-OR implementations [Biamonte04, Biamonte05]. Thus it can be concluded that the AND-EXOR logic has proved to be more useful than AND-OR logic in binary quantum computing, and that when looking for synthesis methods for d-level quantum computing with ancilla qudits we should be influenced by the achievements of the AND/EXOR rather the AND/OR synthesis methods of classical logic CAD. We should be influenced also by layout-related regular arrays of small gates rather than by the large irregular macros that were used by earlier authors [Miller03]. I prove by experimental results of my CAD tools in this dissertation that these ideas of AND-EXOR logic [Sasao93, Sasao97, Salu92] and regularity are very useful. They are, basically the main foundation of the presented work.

Let us observe that, based on the literature the only universal binary quantum gates competitor to Toffoli gates are the Peres and Fredkin gates. Therefore, a natural idea in

MV quantum design is to find the multiple-valued counterparts of all these gates, as well as to find the counterparts of the single-qubit rotations and the counterparts of the Feynman, Controlled-Rotation and Interaction gates. The binary Peres gate has (behaviorally) many EXOR operators inside, so its MV generalizations should have many logic functions that generalize the EXOR operator to multiple-valued logic. These are the modulo-addition operators, for example. Binary Fredkin's internal realization in many quantum technologies is also based on Toffoli gates and Feynman gates, so the generalization(s) of Fredkin's gate should be reducible to our MV-logic-handling methods. There is one exception, however. In some technologies, such as superconducting, the Fredkin gate can be built inexpensively from Square root of Swap gates.

It is interesting to note that the binary AND-EXOR logic represents a special case of Galois Field Logic GF(k) [Batisda84, Stewart89, Winter74, Edward93, Bell66], where the radix k=2 for binary logic. Galois Field logic can be viewed as a generalization of Boolean Logic because the Galois Field mathematical operations are applicable to multi-valued logic values, over any finite field. The use of Galois logic allows us to have a mathematically beautiful theory of such circuits. Therefore Galois Field logics allow us to use mathematics in circuit synthesis algorithms. It is, however, still questionable if this is the best logic from the quantum circuit realization (technology) point of view. This problem is discussed in this dissertation in relation to the proposed d-level Ion Trap realization technologies. In general, however, the selection of the best gates depends on new quantum realization technologies. They appear profusely every year and are hard to

predict. Several technologies and fundamental discoveries in quantum circuits occurred in 2006 – 2009, and the author was not able to incorporate all the newly arriving ideas.

Let us just observe that Galois logic is just one of many possible extensions of the AND/EXOR logic that proved to be dominant in quantum circuits in 2010. The other candidate is the Controlled-Gate-Logic (my own name as this logic is not known in the literature), which seems to be the best option for the current d-level Ion Trap technologies. In our approach we also derive many of new formalisms from the concepts of Galois Field logic.

4.4. Highly Testable Quantum Circuits.

It is well known that AND-EXOR logic, especially the two-level variant of it, is the most highly testable of all digital logic structures [Perkowski97] that are used in classical logic. Observe that the EXOR operator is a (mathematical) group, while the AND and OR operators are not groups. The property of EXOR is that every change of signal in its inputs is transmitted to the outputs. Thus every circuit is more testable when it includes long chains of EXOR gates [Reddy72]. Because quantum circuits have many EXORs, they transmit to their outputs every change in qubits, making them highly testable. High testability of quantum arrays is thus inherited from high testability of standard EXOR/ESOP circuits.

The group property is fundamental to the high testability of EXOR-based circuits and in test vector generation for them. The same is true in d-level circuits - the "group property" of all our logics is good also for the MV case. When I started the research on this thesis in

2002 I believed that the AND-EXOR logic was a good candidate for quantum design, for which the fault-tolerance and high testability issues are especially important [Drechsler99, Kalay99, Perkowski99, Reddy72, Sarabi93]. This belief was also a starting point for the research reported in [Perkowski05, Biamonte05, Pierce05, Allen04], and especially for the work on testable multiple-valued quantum circuits. The top results in testing classical binary ESOP circuits (these circuits include the GRM, FPRM, and PPRM forms as special cases) were given for classical multiple-valued technology by Kalay et al [PHDKalay] as:

…a simple, universal test set which detects all single stuck-at faults in the internal lines and the primary inputs/outputs of the realization… (the) circuit realization requires only two extra inputs for controllability and one extra output for observability. The cardinality of our test set for an n input circuit is (n + 6)…

The test set developed by Kalay et al can also be very successfully applied to Built-in Self-Test (BIST) applications, and the concept of quantum BIST has been developed by Biamonte and Perkowski. However, the Kalay et al methods assumed a stuck-at fault model that is not a good model for quantum circuits in all the technologies that I know.

The quantum circuit testing concepts were developed by Biamonte and Perkowski [Biamonte05] based on classical models [Kalay99, Sasao97, Reddy72]. Based on the fundamental principle of "group gates", they can be generalized to the case of binary oracles realized in d-level Ion Trap quantum circuits from this dissertation using the known "gate insertion/removal" model [Biamonte05] in place of the "stuck-at" model. I believe also that the methods from Kalay et al and Biamonte and Perkowski can be expanded to a wider category of quantum circuits, both binary and multiple-valued, and

not only binary quantum oracles. Further, Kalay et al [Kalay99] have also devised a two-level design method and a minimal-universal test set for the more general Galois Field SOP (GFSOP) form GF(k^r). The GFSOP is analogous to the ESOP as the most general functional representation of its type. Thus, for both the binary and multi-valued logic cases, it is evident that the two-level AND-EXOR logic family is highly testable with a very limited number of test vectors. Again, because of the similarity of these circuits to some quantum circuits, and because of my current understanding of fault models for quantum computing [Biamonte05, Perkowski07], it is highly probable that all their methods can be extended to Galois Field logic.

The highly testable EXOR-logic based circuits that originate from Reddy's seminal 1972 paper were next extended by Sasao [Sasao97] and Kalay et al [Kalay99], and were further extended in classical logic by Bhattacharya et al [Bhattacharya98] to bridging faults and to other AND/EXOR structures. It remains to be investigated if the methods developed for quantum logic by Biamonte and Perkowski [Perkowski07, Biamonte05, Biamonte04] can be further generalized along the lines of the recent works of Bhattacharya, and others.

Concluding this section, currently the basic research and CAD tool development for various binary quantum circuits is increasingly at the forefront of the quantum CAD research as documented by many papers in the Congress on Computational Intelligence, RM, ISCAS, ULSI, ISMVL symposia and DAC conferences [Bullock05, Brylinski01], but the number of papers on d-level design is still less than 20 and most of them come

from the PSU quantum group [Normen07, Khan05, Khan05a]. The literature listed at the end of this dissertation includes the complete bibliography on quantum multiple-valued concepts and circuits as of 2010.

Nowadays, development in the quantum circuit design area is dominated by binary quantum circuit design (with base $|0\rangle$, $|1\rangle$). So far, there has been no work other than a book by Anas Al-Rabadi, based on his Ph.D. from PSU [PHDAl-Rabadi, Al-RabadiBook], that would try to generalize the existing AND/EXOR approaches to d-level quantum circuits. The Al-Rabadi book is already obsolete after a few years since it does not cover the new designs in quantum technologies that appeared after 2002. It should also be noted that most of the design/theory papers on d-level design are very theoretical, and no software tools have been developed for any of these methods.

The situation on MVL quantum in 2010 is thus this – on the one hand, new ternary technologies have proved to be operational, but on the other hand no theory geared towards them exists. Few design algorithms exists, and the methods that exist do not address test issues at all. The research reported in this dissertation thus addresses the lack in this area.

4.5. From binary to multiple-level Quantum Circuits.

The AND-EXOR form has been developed into a complete hierarchy of Reed-Muller (RM) expansions, using the Shannon, Positive Davio, and Negative Davio Expansions, in the works of Sasao [SasaoBook], and especially those by the PSU group [Perkowski91, Perkowski97, Dill97, Dill97a, Schaefer91, Pierzchala94]. This hierarchy is described

with logic equations, forms, trees, and decision diagrams. In my dissertation we will present this hierarchy of binary quantum circuits, both for completeness of the explanation and to introduce my reversible/quantum concepts. Next I will expand them to d-level circuits, and moreover I will add new items to the hierarchy, motivated by their practical applications in d-level quantum circuits. In the quantum interpretation the whole new extended hierarchy obtains a new meaning as a hierarchy of quantum arrays that are relatively easily mapped to the recently proposed Quantum Field Programmable Gate Arrays [Metodi05]. It is also easy to map them to the d-level circuits of my interest. Below we pay special attention to Davio expansions and the related binary Fixed Polarity Reed Muller and Generalized Reed-Muller (GRM) forms, because of their relative simplicity. Our assumption that these forms can be easily generalized to the d-level quantum circuits proved to be true in the course of writing this dissertation.

Although some of the forms from the binary hierarchy have been the focus of logic synthesis and minimization research for many years and by many authors, finding the exact solutions is still very difficult for larger functions, even in the binary case. Therefore, we did not even attempt at the exact minimum solution algorithms, but in our desire to create practical algorithms we concentrated on fast approximate approaches. In the binary case we analyze GRM cascades with ancilla bits, Lattice Diagrams and Nets (all introduced in the following chapters). I also discuss the decomposition of functions to these structures combined with EXOR operators. The (binary) GRM logic is a canonical expression (exhibiting a regular structure), which is a subset of the Exclusive-Or Sum-Of-Products (ESOP) expression. In GRM, for every subset of input variables there exists

at most one term with arbitrary (unconstrained) polarities of variables. (The forms and polarities will be explained in detail in the following chapters). In other words, each term in the GRM is unique in both the variable name and the polarity. It is interesting to note that often the GRM forms may produce results with the number of terms very close to that of the exact minimum ESOPs [Cohn62, Perkowski92, Wu96]. For odd Boolean functions they create only one product term that includes all the variables. This property is advantage of our new circuits over the ESOP circuits for Boolean functions with three to seven variables. The GRM forms are also more easily testable than the general-purpose ESOPs [SasaoBook].

4.6. Some types of multiple-valued Quantum Circuits.

Several multiple-valued logics have been invented for non-quantum circuits. While allowing for greater logic density than that of Boolean Logic, these circuits are still not practical. Although they were used with success in CMOS prototypes, they are not widely applicable in commercial CMOS technologies. There are several speculations, however, that the MV logics will become practical in d-level quantum realization technologies including Ion Trap [Bullock05].

This dissertation is restricted to various variants of the Ion-Trap quantum technologies taken from the literature and not invented by me. Not being a physicist myself, I have to rely on authorities with respect to the future prospective proposed quantum technology and in recent years the Ion Trap is the most dominating future technology, of quantum computing. This opinion was expressed by many top experts in the field; Isaac Chuang –

the creator of the first 7-bit NMR quantum computer and Mark Oskin – one of inventors of Ion Trap quantum architecture.

Quantum technology offers an entirely new prospect for computing, and does not merely speeds-up the current computing model. This technology also allows for uniform realization of binary and multiple-valued circuits. Chuang, Chong, Oskin, Bullock and other authors of aforementioned papers are Physicists or Mathematicians, while Dr. Perkowski looks to circuits from a CAD tools perspective, the point of view that I take in this dissertation as well.

The main question is "how to realize multiple-valued logic efficiently using the existing quantum realization Ion Trap technology" The first idea is to adapt the existing MV-logic methods from CAD tools for classical realization technologies such as CMOS or bipolar. Another idea is to use optical computing concepts to develop new algorithms. As an example, one method of implementing multiple-valued logic is with the two-argument Min and ModSum operations, the so-called MIN-MODSUM logic. Dueck and Miller [Dueck86] presented a four-valued CCD PLA utilizing the Min and ModSum operations in place of the traditional binary AND-OR logic planes. Because the newly proposed QFPGAs are based on CCD-like models of logic, this dissertation may become quite practical for the forthcoming quantum technologies.

In their research, Dueck and Miller showed that:

> *... for one variable and (two variable) latin square functions, the MODSUM always results in a PLA with no more product terms*

than a PLA using either the truncated sum (SUM) or the MAX and typically requires fewer. Despite the higher unit cost of the MODSUM, the reduced number of product terms usually results in a lower overall cost.

However, even for the four-valued logic presented, the circuit is much more complex and requires more gates than that of its Galois counterpart (with both the addition and multiplication operations) that will be presented later in my dissertation. A realization that is nonsensically complex in the contemporary CMOS-based multiple-valued circuits, may be the best choice in quantum technologies in which there are no natural, physics-based constraints on the radix of logic (two values is the practical constraint of CMOS). Therefore, several approaches to build multi-valued quantum circuits have been proposed [Muthukrishnan00, Kerntopf04, Khan04a], and still proliferate in the recent literature. Some gates have approximately the same cost in both technologies, Inverter is one example. The quantum Inverter requires only one Pauli rotation [Nielsen00], so it is thus the cheapest possible quantum gate. Inserting inverters to any qubit in a circuit is also not a problem in any quantum layout type that I am familiar with. On the other hand, some gates are inexpensive in quantum logic when compared to classical logic. The best example of this cost trade-off between classical and quantum circuits is the Hadamard transform. A one-bit Hadamard transform requires only two single-qubit Rotation gates internally, so it is the cheapest "quantum gate" (after the inverter gate) in quantum design. This gate causes the quantum circuit to operate in the spectral domain, which is expensive in classical logic. On the other hand in classical logic design the Hadamard transform uses one multi-bit subtractor circuit and one multi-bit adder circuit, being thus

a big and complex block using many AND, OR and EXOR gates. Therefore the Hadamard Transform for many variables is very expensive in the classical hardware world. However, in the quantum world, the Hadamard Transform costs practically nothing, since the butterfly of the transform (corresponding to the Kronecker product of single qubit Hadamard gate) is built for free by the tensor product of parallel quantum systems. The Hadamard Transform is the perfect example of using the power of quantum computing in calculations. The Hadamard gate should be the basis for designing those types of quantum circuits that would not just mimic classical circuits. (This gate is the basis of the famous Grover algorithm, one of the two main quantum algorithms). One has to look for more gates like these, inexpensive but powerful in quantum design.

Although for other quantum gates the differences are not as dramatic as for the Hadamard gate, the problem is very characteristic when comparing quantum and classical circuit design: "*what is cheap in classical logic may be very expensive in quantum logic (like the OR of many terms) and what is expensive in classical logic may be very inexpensive in quantum logic (like Hadamard)".* This is an important observation that explains why the entire design methodology using quantum gates should be deeply reworked. I believe that methods may be adapted in quantum design from the classical design only with deep care. I conclude that my methods should use Hadamard, Inverter, Pauli rotations, rotations, and CNOT gates as much as possible as these are cheap gates in Ion Trap technology. The gates should be realized only on neighbor qubits, again to reduce the costs in linear Ion Trap technology.

The research of Dueck and Miller on MIN-MODSUM circuits does not present functions with more than three variables. This is a useful condition for quantum circuits. But possibly this is a limitation of the PLA technology of the era when their paper was published. As such, no practical applications were discussed by them since realizing the multi-input EXORs and MODSUMs is difficult in all classical technologies (it is however cheap in Ion Trap technology). Again, it is in contrast with the quantum world where the EXORs and MODSUMs are two of the few least expensive gates [Giesecke07]. (Practical ternary/binary quantum circuits are discussed in [Khan05, Khan05a, Perkowski06].) Further, it is noted within Dueck and Miller's research that "*minimizing an expression using the MODSUM… appears to be a difficult problem*" and is not addressed. In contrast, the minimization of multi-valued logic implementations with AND-EXOR circuits is very efficiently addressed by the EXORCISM-MV-2 minimization software that was developed in our research group at PSU. (EXORCISM-MV-2 is a software package providing "efficient minimization" of arbitrary ESOP expressions for multiple-output, multiple-valued input, incompletely specified functions [Song93]). The issue of testability for multi-valued generalized Reed-Muller circuits in modulo m sum-of-product form was investigated by Dubrova and Muzio [Dubrova96] (with m being a prime number greater than two). In their research, both the number of tests required for fault detection and the generation of the tests were addressed. It was found that:

Just four tests are sufficient to detect all single stuck-at faults on internal lines in the circuit. Furthermore, this set of tests is independent of the function being realized and

therefore universal. We give two alternative techniques for testing primary inputs – one by generating a test set of maximum length 2n, where n is the number of primary inputs, and the other by adding to the circuit an extra multiplication mod m gate with an observable output to ensure that the four tests for internal lines also detect all single stuck-at faults on primary inputs.

Hence, for the Galois Field PPRM (GFPPRM) form, Dubrova and Muzio [Dubrova96] have developed a universal (possibly minimal) test set for prime cardinality fields. In contrast, Kalay et al [Kalay99] have developed a two-level design and minimal-universal test set for the more general Galois Field SOP (GFSOP) form. Here the Galois Field is not limited, as it is of size GF(k^r), and the SOP form is unrestricted by polarities or terms. Such circuits may be implemented in current CMOS technologies [Zilic93, Pierzchala94], but their real advantage will be demonstrated in quantum Ion Trap technology [Muthukrishan00]. This technique may also be applicable to Galois Logic. Finally, the most significant difference between the Min ModSum and Galois Logics is that while the former is simply an attempt to extend the high density and good testability characteristics of the Reed-Muller Expansion to the multi-valued case, the latter presents a new multi-valued algebra in which an entire hierarchical family of trees, decision diagrams, and forms can be elegantly developed. Therefore, our approach generalizes the researches of Hurst and Sasao and the Green/Sasao hierarchy. [Hurst85, SasaoBook]. At this stage, I believe that all the advantages of Ring Logic and Min-Modsum Logic can also be found in Galois Field logic, but I will further investigate this topic after completing my dissertation.

Boolean AND-EXOR logic is a special case of GF Logic, for a field of size 2. Because Galois Field Logic, for both the binary and multi-valued case, exhibits the group property, it is highly testable. Observe that the group property is closely related to the reversibility that itself characterizes all quantum circuits. AND-EXOR (Boolean Logic) forms have already been demonstrated to be highly testable [Pradhan78, Reddy72, Sasao97]. Since the Galois Field addition operation is the generation of the EXOR operation, I believe that the known testing and error detection techniques can be naturally extended from Boolean Logic to the quantum multiple-valued Galois Field Logic. This belief was also based on the fact that the high testability of classical multi-valued logic GF circuits has also been demonstrated by Kalay et al and Dubrova et al [Kalay99, Dubrova96].

To incorporate the multi-valued logic method in binary VLSI design and new types of FPGAs, the Reed-Muller form was extended for Galois Fields [Stankovic97]. However, as mentioned above, there exists a strong link between the Reed-Muller and Galois logics and quantum technology. This link is even stronger than that between these logics and any other existing technology. Hence expansion types such as GF-Shannon and GF-Davio exist, which can be utilized to derive all other hierarchical forms (i.e. GF-KRO, GF-PSKRO, GF-FKRO, etc.). Galois Theory demonstrates that for every number k^n, where k is prime and $n \geq 2$, there exists a unique matrix under mathematical operators, such that all group theory properties are satisfied. Thus, it was again demonstrated that the technique that can be utilized for the representation of binary logic, can also be

extended to multi-valued logic. Our research specifies the relationship between the Reed-Muller Hierarchy and the Galois Fields. It provides the capability of expressing multi-valued logic with decision diagrams and compact algebraic structures (or forms), which can be represented with "universal" expansion circuits. Such circuits are next mapped to structural binary technology [Tsai96]. Galois Fields of size 2^n can be interpreted for a binary hardware implementation. It can be also shown that for GF(4) and GF(8) the field operations of both addition and multiplication can be performed with few ordinary binary logic gates [Dill97]. We expect therefore that they can also be built relatively easily using single qubit operations and interaction gates.

4.7. Heuristic search methods for Synthesizing Quantum Circuits from the above-introduced gates and circuits.

Much of the research described in this dissertation was undertaken in order to develop algorithmic approaches to designing application-specific quantum algorithms for selected classes of multiple-valued logic synthesis and minimization problems. They are included in the following chapters of this dissertation. However, in order to understand these approaches, much background on classes of functions and design methods for them will first be explained to make this thesis self-contained and comprehensive. Several other new approaches are developed in this dissertation to utilize search techniques for logic minimization of d-level quantum circuits.

Several new methods have been developed for regular structures in chapter 5. They lead to a complete detailed synthesis algorithm (Chapter 6) that was programmed, tested and evaluated on many benchmarks. This algorithm is a representative of a family of

algorithms related to the design of quantum circuits with ancilla bits for Kronecker Lattice Diagrams (satisfying 2D neighborhood constraints). Next, the minimization of multiple-valued generalizations of such diagrams are discussed. Although these methods require relatively high numbers of ancilla bits, they use only quantum realizable 3×3 Toffoli gates, thus our method reduces the costs of large functions. The research from [Dill97, Giesecke06] used standard (Darwinian) evolution as a model for which logic minimization algorithms for quantum arrays without ancilla bits were determined. Till now, the few exact minimization algorithms developed have required nearly exhaustive searches on standard computers and are quite time consuming. I judge this model to be insufficient, and thus the ideas of search and heuristics were added. Adding ancilla bits also makes these circuits more similar to classical circuits and should make our methods more successful for practical circuits (benchmarks).

Next, the above basic methods were extended to ternary logic as an example of multiple-valued logic. The goal of using our new approaches for d-level quantum circuit synthesis and optimization in this dissertation was to create non-exact fast approximate minimization techniques that would be an improvement to the methods proposed in the literature. The minimization techniques developed in this dissertation are in principle applicable to both single-output and multiple-output MV quantum circuits, binary, ternary and quaternary, but the software was developed only for the binary case.

Several variants of Genetic Algorithms and Genetic Programming were used at PSU to minimize FPRM circuits [Dill97, Dill97a, Dill98, Zeng95, Muller94, Drechsler95] and

various types of reversible circuits with general structures. For instance, several attempts were made by Karen Dill and Marek Perkowski in years 1996–2005 to develop a purely evolutionary (i.e., GA with no human-designed heuristics) approach to the minimization of GRM forms. As no application-specific knowledge was incorporated for these designs, the results were remarkable as they compared favorably with those of the heuristic algorithms designed by human experts [Debnath95, Debnath96]. On the other hand, for some functions, Sasao and Debnath found better solutions using a heuristic knowledge-based algorithm, which showed that the evolutionary approach should perhaps be equipped with some deeper human-like knowledge, and/or that some human intervention in the automatic solution process should also be incorporated. Although Debnath and Sasao developed a successful heuristic for GRM minimization, capable of handling functions with a large number of variables and multi-outputs, their software was not available in the public domain. An automated technique for logic synthesis and minimization for incompletely specified data in GRM form was developed subsequently by Karen Dill; the iGRMMIN Software. Finally, the AffineGenerator and ECPS software tools were created by Sazzad Hossain [PHDSazzad] but applied only to binary circuits.

It is important to note that while the first GA-based algorithms did not guarantee to find a correct solution, the new wave of GA algorithms, based on polarities [PHDSazzad] and on the input variable ordering (this thesis), always give a correct solution, and use the evolutionary principles only to optimize its cost. Every circuit found is correct, and the evolutionary/search principles are used only to optimize the circuit. As we shall see,

when more advanced algorithms are developed, the role of evolutionary ideas in them decreases.

4.8. Heuristic Search versus Nature-inspired Evolutionary Search.

Several concepts contributed to the research presented in this dissertation. My work is based on 22 years of experience of the PSU group in AND/EXOR logic and reversible design as well as on recent papers from other groups. I am aware of the results on evolutionary research from [Dill97b, Dill98], but using only the heuristic search approaches. My approach aids humans in designing efficient algorithms for binary multi-valued quantum circuit synthesis and optimization problems. This approach uses classical search methods with some probabilistic methods – the problem formulation, its cost function, constraints, heuristics, etc are more important than the final representation of the search in one or another software. Our fundamental approach starts from the assumption that any problem in our class can be solved by searching some space of known states (for instance, these states are the circuit polarities or the circuit structures being optimized). Solutions in this new approach are achieved with an intelligent strategy using technology constraints, human-designed heuristics and state-space search mechanisms [Nilsson98, Luger02], that are different from the standard evolutionary algorithms such as GA or GP, and also different from the well-known quantum inspired evolutionary algorithms [Kim00].

Traditional exhaustive search mechanisms (breadth first, depth first, branch-and-bound, etc.) give assurance of finding the exact optimum solution in the solution space, but are (often) prohibitively time consuming [Giesecke06]. Thus, both complete (searching the

entire state-space) and incomplete (evolutionary and rule-based) search strategies may be unsatisfactory for producing an efficient and effective problem solving technique for our class of applications. In contrast, our new approach incorporates new representation and heuristic search strategies and problem solving/learning paradigms into a synergetic system.

Concluding remarks on the main philosophy of this dissertation:

(1) A recently proposed breakthrough new Ion Trap quantum technology of d-level quantum circuits has been used, for which our circuits are optimized. This makes our research very up-to-date since nothing was published in 2010 or earlier about CAD for this technology.

(2) A powerful multi-valued algebra has been invented to synthesize circuits optimized for realistic cost functions. It combines the properties of linear independence, Galois Fields, and the Reed-Muller logic hierarchy.

(3) The usefulness of this algebra is illustrated on several practical circuits realized for the selected technology models.

(4) A number of synthesis methods and techniques are invented and realized in a new type of variable–ordering meta-algorithm that was ultimately invented. This approach creates a general framework of synthesizing regular structures in which a single method is used for practical logic synthesis and minimization.

(5) Further, an analogy and extension of the binary AND/EXOR logic to the new Galosi Logic is made. Hence, the usefulness of the Galois Logic for multi-

valued logic synthesis of d-level quantum systems created here is demonstrated.

(6) The concepts of regular reversible structures based on mapping logic circuits to one dimensional and two dimensional Ion Trap quantum layouts are introduced and illustrated. These concepts are for various Ion Trap circuits, both binary, multiple-valued and hybrid.

(7) Binary quantum oracles can be built using several synthesis methods developed in this dissertation. This can be used in the generalized Grover algorithm that uses oracles with multiple-valued inputs and binary output. Because many methods are available now, a better oracle can be built by an amalgamation of methods.

(8) Some cascade synthesis and Lattice constructing software has already been implemented and discussed in full detail. Much more software may be developed based directly on other ideas presented in this dissertation.

4.9. Introductory illustration of some basic ideas of this dissertation.

The ideas presented in this thesis are many and complex. Therefore, to help the reader of this dissertation, in this section I put together some of my main ideas – research questions that I want to pursue in the next chapters of the dissertation in full detail, and illustrate them with examples.

The first observation is that other authors create all their algorithms for a circuit model that is not realistic, as it assumes that there can be a gate located between any two qubits, even if these qubits are located far away in space from one another (for instance in 1D

Ion Trap). This assumption may be sufficient for very small circuits but is absolutely unacceptable for the big quantum circuits of the future. The gate between any two qubits would mean an immediate direct interaction between any two ions in the Ion Trap, which is physically not possible. As we remember form Chapter 2, the ions can be stable or flying (teleported). Let us concentrate for a moment on stable ions. In the simplest (but practical as of 2010) case, all ions in the Ion Trap are placed linearly (as a vector). Thus every ion (qubit) can interact at most to one neighbor above and one neighbor below. This physical constraint of "2-neighbor" quantum layout of the substrate has much influence on practical designs. As an example, consider the very simple 4×4 Toffoli gate shown as a unit in Figure 4.1a. Other authors (MMD (Miller, Maslov, Dueck) or Agrawal and Jha) leave such gate as a final result of synthesis with cost 1 (or cost 5, which comes from 3 inputs in the product). This is not accurate and is simply wrong! As we see, to realize this circuit in quantum, one ancilla bit should be added as in Figure 4.1b. Next, each of the 3×3 (standard) Toffoli gates from Figure 4.1b are macro-generated to the so-called "Barenco circuits" [Barenco95], thus creating the quantum array in Figure 4.1c. This would be fine if every two qubits could interact directly – but they cannot. So we have to use transformations from Figure 4.1d to create 2-neighbor-only type of circuits.

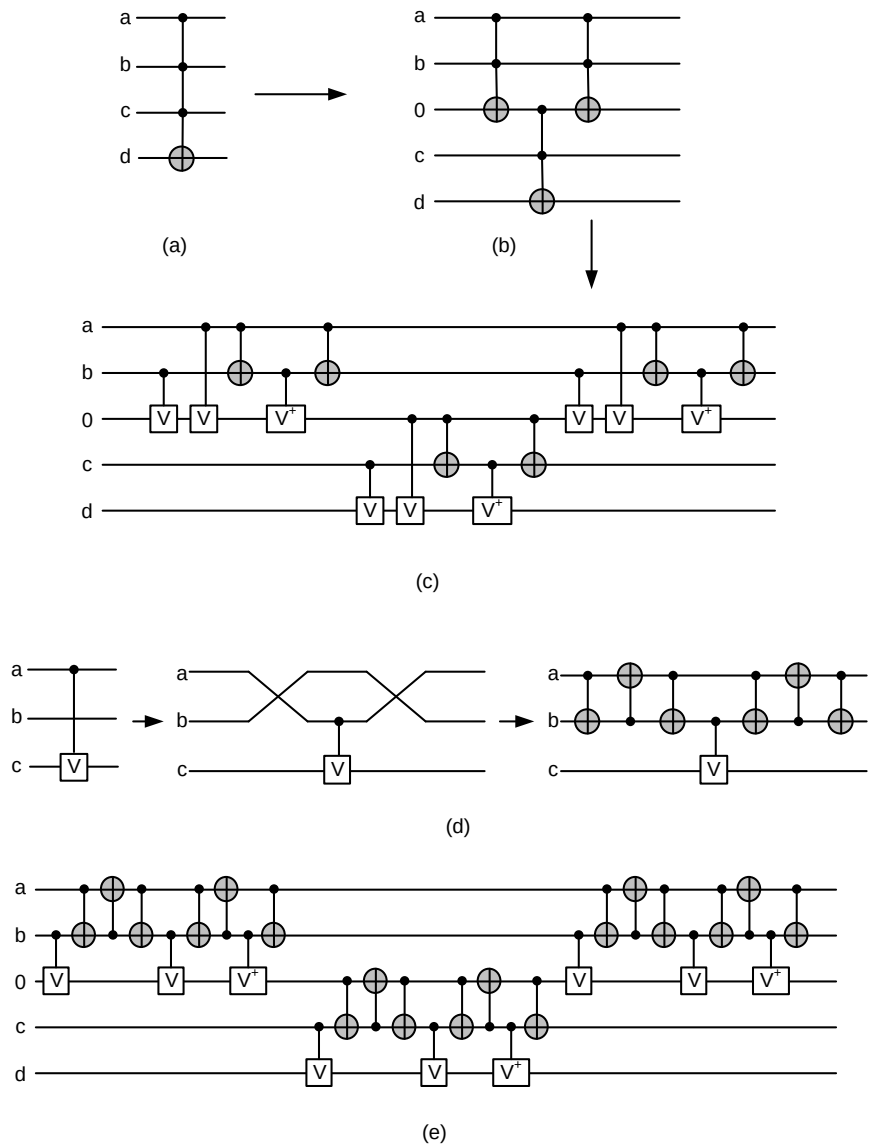
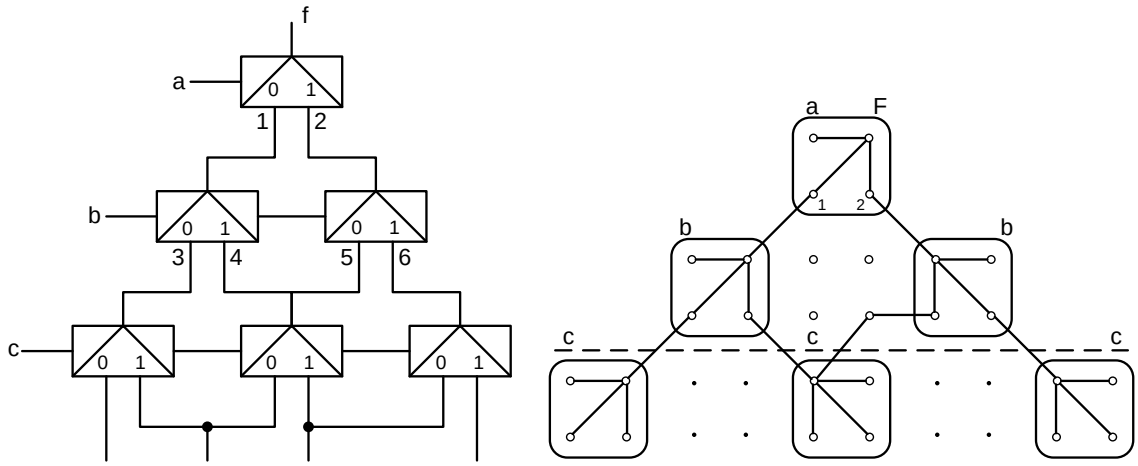


Figure 4.1. This example illustrates the nature of a problem with the linear Ion Trap. A 4×4 *Toffoli gate that looks like a cheap gate is, however, quite expensive when mapped to linear-neighborhood quantum array. (a) symbol of a gate as used by other authors, (b) its decomposition to Toffoli gates using one ancilla bit, (c) the final circuit with 2-qubit quantum primitives, but not-realizable in linear neighborhood as it has wires going over gates, (d) steps to realize the gate with a wire going over it, (e) the final circuit in linear neighborhood Ion Trap for the initial gate from Figure 4.1a. The number of gates in this circuit reflects the true quantum cost of the gate (macro) from Figure 4.1a.*

The final circuit for the gate from Figure 4.1a is then shown in Figure 4.1e. It has 27 2×2 gates in 2-neighbors-only topology. My dissertation takes these kinds of practical considerations into account. This synthesis approach is formulated for the first time, and

although it has perhaps no special influence for small Ion Trap circuits build in 2010, it will significantly affect the architectures of future Ion Trap Quantum Computers which will include large quantum circuits.



(a) (b)

Figure 4.2. Example of Lattice diagram realization in quantum. (a) Schematic of a Lattice diagram using standard multiplexers, (b) First attempt to map the schematic to 4×4 *grid of 2D quantum layout. Each dot represents an ion and each oval rectangle represents the (time evolving) internal qubits of the multiplexers. Although it looks like fan-out of the middle bottom mux is two, it really is one thanks to go-through signals in some gates. Ancilla bits are not shown. This method can also be applied to the new types of reversible lattices introduced in next chapters.*

Based on the above example, we see that the quantum circuits should have short connections. As discussed in [Perkowski97, Jeske98, Shah00] short connections require regular structures. There are two main regular structures in the literature: Lattices [Perkowski97a, Perkowski98] created by adaptation and generalization of Akers Arrays [Akers72], and Nets [Perkowski02, Al-Rabadi02] that are original designs for binary logic. They are both two-dimensional and thus are expected to map well to 2D Ion Traps. First I tried to analyze the mapping of Lattice to 1D quantum layout (as in standard quantum arrays). I found (Chapter 5) that the internal connections of the Lattice map

well, i.e. with small distances, but there is a big problem with connecting all multiplexers to input variables – this involves very many SWAP gates. Hopefully if we assume 2D quantum layout in 4×4 grid, with 4 neighbors of every qubit (dot in Figure 4.2b), then the layout dramatically simplifies under the assumption that all variable inputs can be individually accessed or that they run in buses. The initial placement is presented in Figure 4.2b. Figure 4.2b assumes that all variable inputs are individually initialized by electromagnetic pulses, given in series or in parallel. Long enough coherence of all ions is assumed. This is only one example of a Lattice circuit realized in Ion Trap technology; actually I will show in the next chapters that other lattices are better for quantum realization. This example was just to explain the problem on a well-known circuit [Perkowski97].

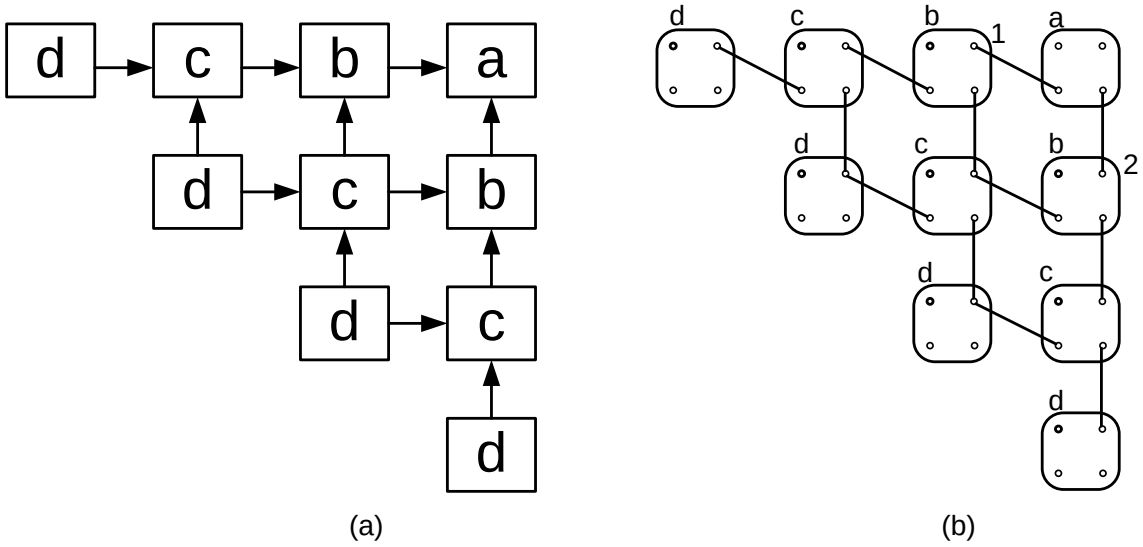


Figure 4.3. Lattice diagram realization in 2D Ion Trap. (a) The abstract grid with cells, (b) Its placement assuming individual (serial or parallel) access (control) of all input variables corresponding to individual qubits. Sufficiently high decoherence (typical for an Ion Trap) is assumed. Mapping of input variables is not shown in (a) or in (b).

Figure 4.3 assumes that connection between any two dots inside every square is possible (this is achieved using methods from Figure 4.1). We are not specifying the nature of the

logic in the cells. As we will see, various gates can be used inside. External connections to neighbor cells (rectangles) are shown and they exhibit perfect regularity. This mapping is of course scalable to any size array (assuming no physical constraint on the size of 2D arrays of qubits – ions, atoms, etc). Figure 4.4 shows the initial layout of input buses for the lattice, and Figure 4.5 presents an improved layout of input buses. The realization of a 2×2 qubit cell for a multiplexer block is presented in Figure 4.6. Detailed layouts like this for various types of lattices and with different designs of internal gates are constructed and compared in this dissertation.

Now, let us analyze another regular structure, the Net, which combines a triangle of gates to generate some (sufficient) subset S1 of symmetric functions with a rectangle of linear functions that creates the final vector of symmetric functions by EXOR-ing subsets of functions from S1 (Figure 4.7). Assuming a linear layout of qubits, we create the quantum array from Figure 4.8. Unfortunately, as the Net grows to a larger number of input variables, the distances between the control and target qubits in CNOTs grow (up to 5 in Figure 4.8), which demonstrates that this architecture is not well scalable for the linear Ion Trap layout.

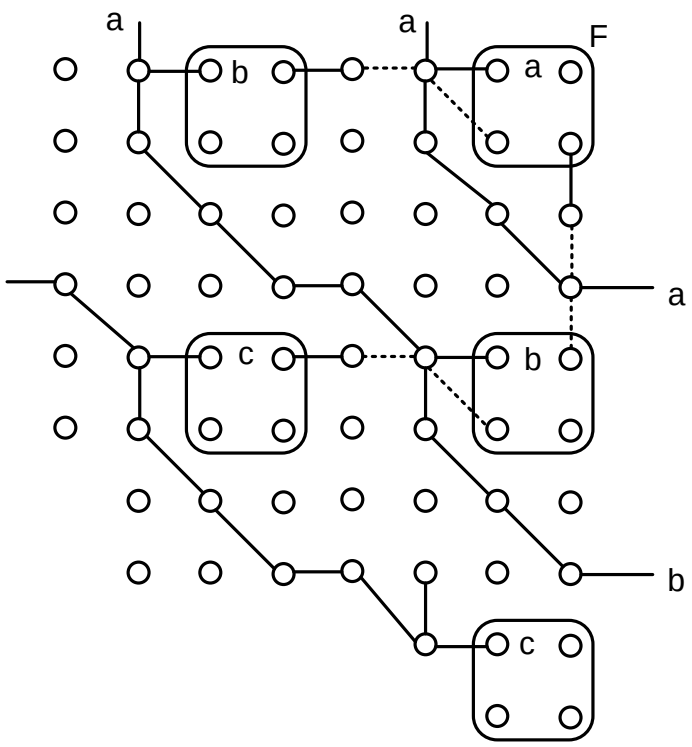


Figure 4.4. Initial lattice layout for 2D Ion Trap technology with width 2 of channels for buses.

Now I will present an idea related not to Ion Trap quantum technology but to algorithm design. Evolutionary algorithms like GA work in an unsorted (non-oriented) space of genotypes. These algorithms in some cases may miss a solution chromosome even being very close to it in the sense of Hamming Distance. Systematic exhaustive search methods search the entire space, which takes too much time and computer memory space. Heuristic tree search methods have an advantage over GA or exhaustive search only when a good selection heuristic (such as quality function minimization) is known and applied. The question that arises is therefore this:

"How can we search the space of all solutions, if we do not want to "nearly randomly" jump from one element to another (as in a GA), and we do not want to systematically go

to every element (exhaustive search) either, but we believe that there are some regions of this space that their elements (solutions) have better costs?"

This search idea can be visualized as a rugged 3D terrain of cost function with the mountain tops corresponding to the best cost function values. We want to use partial information and expect that close to good solutions, some even better solutions exist. Here we can apply the general optimization methods known from the area of heuristics. The space with metrics must be available in such methods. Such a metric space does not exist in standard GA algorithms.

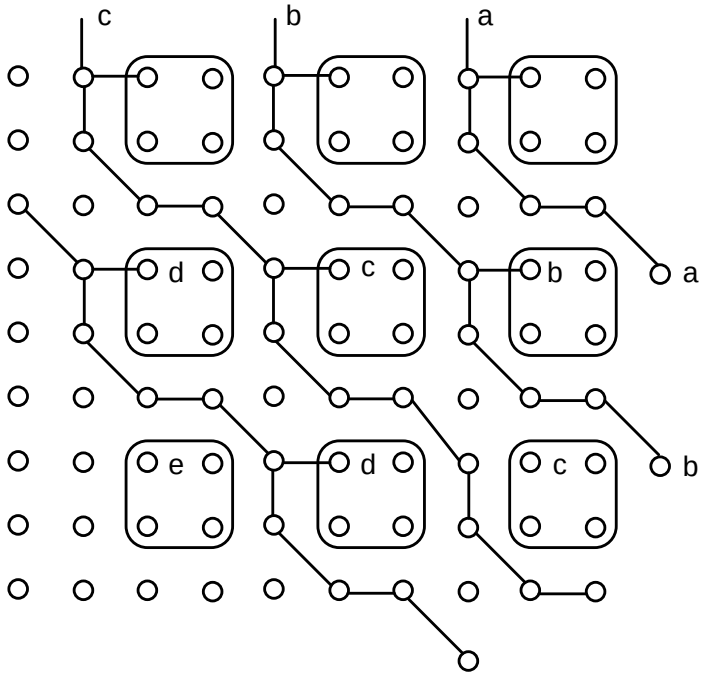


Figure 4.5. Improved layout for a lattice with width 1 of channels. Each dot is an ion. We have to analyze if some future improvements can be done to this layout. My goal is to improve this circuit, making it more compact by some small overlap of connections and logic in cells.

We have to know the structure of the search space: which elements are in Hamming Distance 1 from one another, which elements are in distance 2, etc. We need to create a

certain distance matrix for all elements of this space. Once we generate this matrix, we can use one of many metric-space-based methods.

Observe that in contrast to the evolutionary algorithms presented by Lukac and other authors, where non-solutions are generated most of the time, when our algorithms build various types of Lattices, Decision Diagrams, Nets or Decision Trees every constructed circuit is correct. However, as verified in my preliminary programs, the quantum costs of circuits for various orderings of variables can differ considerably. We want to find an element in the space of all variable orderings for which the cost is minimum or close to minimum. This principle allows us to use various costs and the ordering algorithm is the same for every ordering, it only uses a different cost function. This is an advantage in a situation where many variants will be created and compared.

Based on the above principles, we want to find a single permutation in the space of all permutations which has the minimum (quantum) cost. The space of all permutations is presented in Figure 4.10. It is a Lattice, where every element is a permutation of elements 1, 2, 3, 4. (This "Lattice" concept is of course different from lattices which are types of quantum circuit structures and lattices used in quantum physics.) Two elements (nodes) of the lattice are linked by an edge if they differ by one swap of neighbor characters in strings.

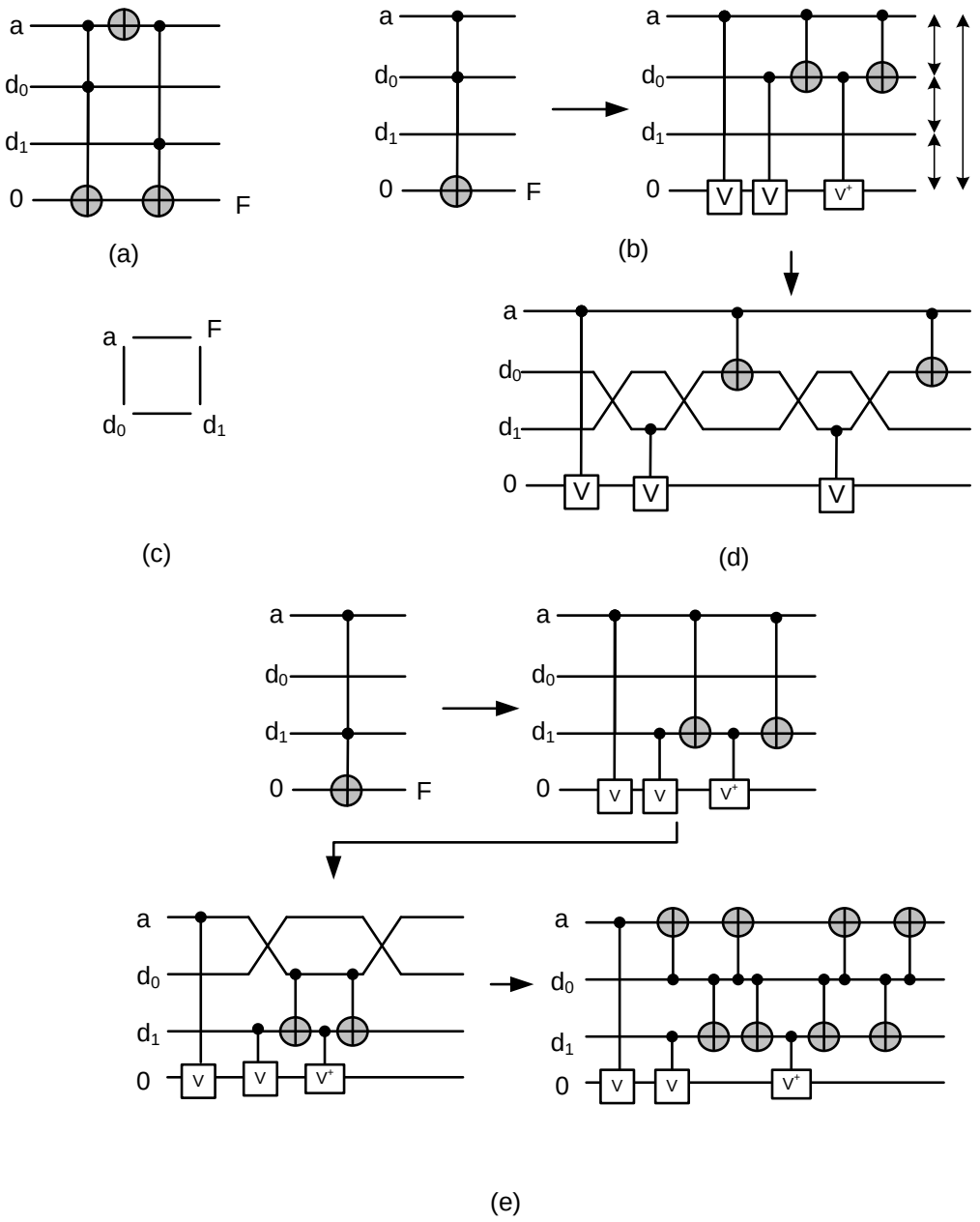


Figure 4.6. The realization of mux on 4 qubits with 4×4 *grid. (a) classical multiplexer realized in quantum array with one ancilla qubit, (b) realization of the first Toffoli from Figure 4.6a, (c) the neighborhood for the ions realizing the mux from Figure 4.6a, (d) mapping of Figure 4.6b with respect to the neighborhood pattern from Figure 4.6c, (e) mapping of the second Toffoli gate from Figure 4.6a to neighborhood pattern from Figure 4.6c.*

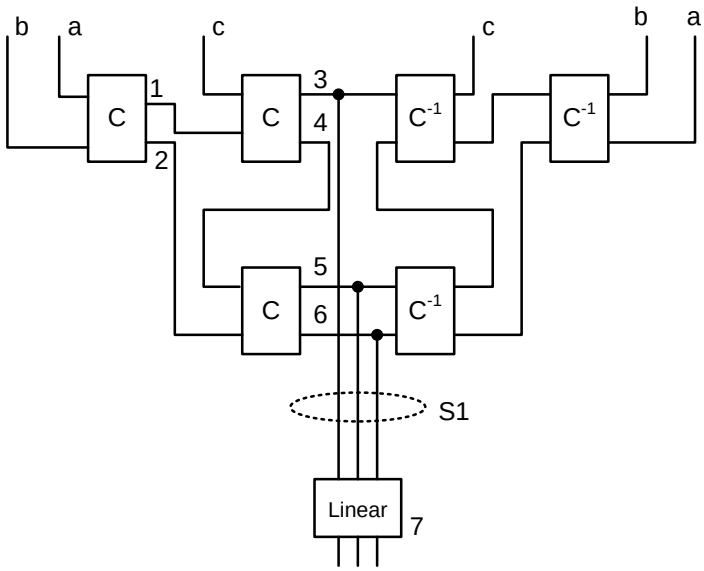


Figure 4.7. Reversible Net structure to generate all multi-output symmetric functions of variables a, b, c. S1 is the net of symmetric indices from which all symmetric functions can be created. Block C^{-1} is the inverse (mirror) of block C.

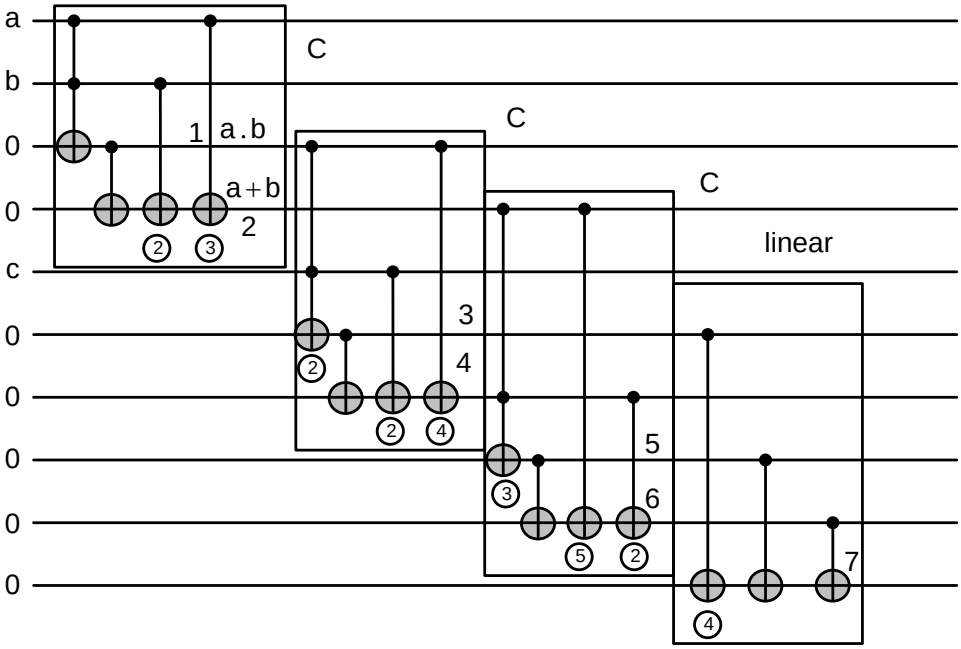


Figure 4.8. Standard quantum array (with dimension of time from left to right) for part of the reversible Net Figure 4.7. Observe here the unfortunate lengthening of connections between qubits. The members below the EXOR gates show the maximum distance in all Toffoli and Feynman gates. The block C is shown, the mirror circuit with C^{-1} is not shown.

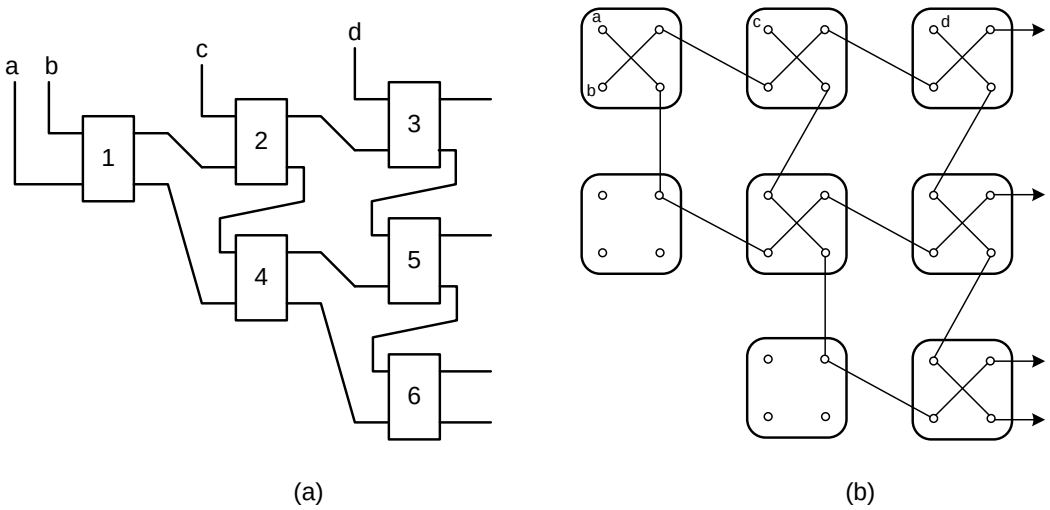


Figure 4.9. Realization of regular structures called Nets in 2D quantum layout. (a) The general structure with regular and short connections, (b) Its mapping to 2D 4×4 *grid or* 8×8 *grid of ions. There is no need for buses as inputs are initialized only once on the top.*

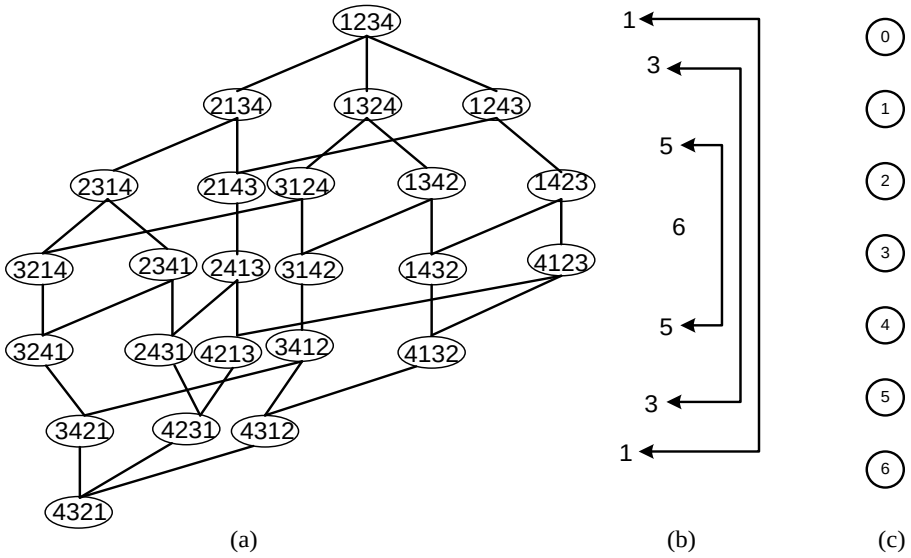


Figure 4.10. Lattice of all permutations of elements {1, 2, 3, 4}. It is used as a space of all orders of expansion variables for decision trees, decision diagrams and Net structures in optimization algorithms based on Nature. (a) the lattice in which strings in levels correspond to single transpositions (swaps) of their parent strings, (b) illustration of symmetry and numbers of elements in levels of the lattice, (c) the distance of the levels of the lattice form the initial permutation string 1234.

4.10. Conclusions.

Within this chapter the characteristics of known group-based logic families have been reviewed and new logic families of their Galois based counterparts have been developed as an original contribution of this dissertation. This invention process will be shown in more detail in next three chapters, and several examples will be given. It will be shown in a systematic way that given an algebra with mathematical operations constituting a complete logic system, providing the capability to construct linearly independent functions, there are a number of ways to create a universal module and the corresponding expansions. The AND-EXOR Sum-of-Products algebra is selected here, chosen for the high density, good testability, and structural regularity of circuits created with this algebra. The logical family hierarchies are then developed starting from expansions. Trees, decision diagrams, and two-level forms then complete the logical hierarchy development, and will be given in subsequent chapters. This technique for the invention of new algebras is applicable to any group-based logic. Possible binary implementations for the multi-valued logic designs are also investigated.

Of course, the work outlined in this chapter is just a first step. It is much extended in the dissertation, chapters 5 - 9. The following issues need to be addressed:

(1) Algorithms to find the best trees, diagrams, and forms.

(2) Realization of practical binary and multi-valued circuits of the new universal modules and Galois Field gates.

(3) Benchmarking functions to compare the new solutions to well-known solutions based on classical binary and multi-valued synthesis methods.

These are done for some selected structure and algorithm variants in chapters 5, 6 and 7 of this dissertation.

CHAPTER 5

Regularity and Ancilla Bits in Binary Quantum Circuits.

As we remember from section 4.9 in the last chapter, regularity of design based on short connections is fundamental to realizing quantum circuits in both linear (1D) and 2D Ion Trap technologies. On the other hand, trees (which may be regular only when small) and diagrams (usually not regular) have been introduced in classical binary logic. The ideas of expanding and mapping of logic functions to regular diagrams will be extended and combined for binary logic in this chapter, and generalized to ternary and quaternary logic in chapters 8 and 9.

5.1. Some preliminary ideas.

The well known AND-EXOR hierarchy has been designed in [Perkowski89], extended by Sasao in [Sasao91a] and further extended [Perkowski97d, PHDSazzad], Chapter 3. I will use some concepts from these hierarchies. As we know from Chapters 3 and 4, the concepts of logic expansions and trees are fundamental to classical logic synthesis. The concepts of Shannon expansions and the corresponding trees (Figure 5.1.1), as well as Kronecker trees (Figure 5.1.2), Pseudo Kronecker trees (Figure 5.1.3), and Positive Davio trees (Figure 5.1.4) based on Davio expansions have been developed [Drechsler94], and are much used in classical logic synthesis theory [Stankovic97]. All these trees can be realized in regular layouts to a certain size only, as the trees in general grow exponentially from level to level. The trees are, however, the base of all our new concepts that attempt to achieve regularity of applying expansions in a geometrical space.

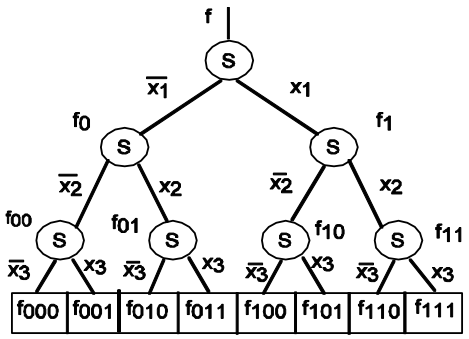


Figure 5.1.1 Shannon Tree

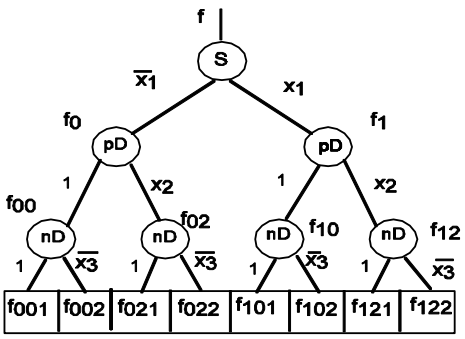


Figure 5.1.2 Kronecker Tree

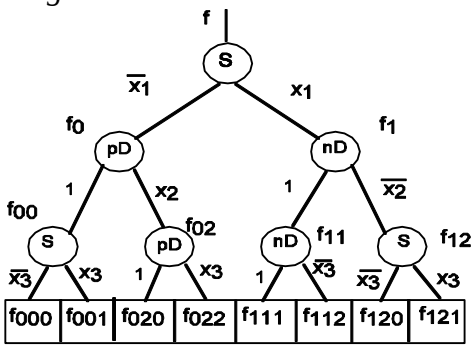


Figure 5.1.3 Pseudo-Kronecker Tree

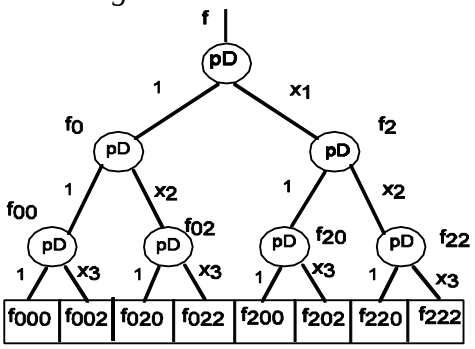


Figure 5.1.4 Positive Davio Tree

The Figures 5.1.1 – 5.1.4 present examples of regular diagrams. I will present below other regular diagrams from the literature, and I will invent new regular diagrams.

Regular diagrams are mapped to *regular structures*, also called regular substrates, regular layouts, grids, or regular connection patterns. Other names are also used for them in the literature. The Figure 5.1.5 shows some examples of regular layouts. The mappings are "into", which means that not all neighbor nodes and edges of the grid must be used, but every node and edge of the regular diagram must be mapped. The most well known regular diagrams are binary balanced trees and lattice diagrams.

5.2. From binary function expansions to trees and lattices.

5.2.1. Combining binary trees and binary lattices in regular layouts.

The foundation of my approach to **regularity-geometry-driven structure design of combinational quantum circuits** is **expansion** of functions, i.e. operators that transform a function to few simpler functions. I restrict our attention in this dissertation to combinational circuits (no memory). As we remember, in a well-known canonical Shannon expansion, function **f** is expanded with respect to input variable **a** as follows:

$$f = af_a + \bar{a}f_{\bar{a}} = \; f = af_a \oplus \bar{a}f_{\bar{a}} \quad = {}^{1}\mathrm{a}\ \mathrm{f}_{1\mathrm{a}} \oplus \; {}^{0}\mathrm{a}\ \mathrm{f}_{0\mathrm{a}} \quad \text{(Equation 5.2.1)}$$

where $f_a = f\big|_{a=1}$ and $f_{\bar{a}} = f\big|_{a=0}$ are the positive and negative cofactors of function *f* with respect to the variable *a*, respectively. Here ^{1}a and ^{0}a are Post Literals (we keep this notation in MV logic). In case of binary logic, other possible expansions include only the Positive Davio and Negative Davio, but already in the ternary radix of logic the number of expansions is very high, and in the quaternary radix it becomes astronomical. In trees and diagrams, as we remember, the application of these three expansions is very similar (Chapter 3).

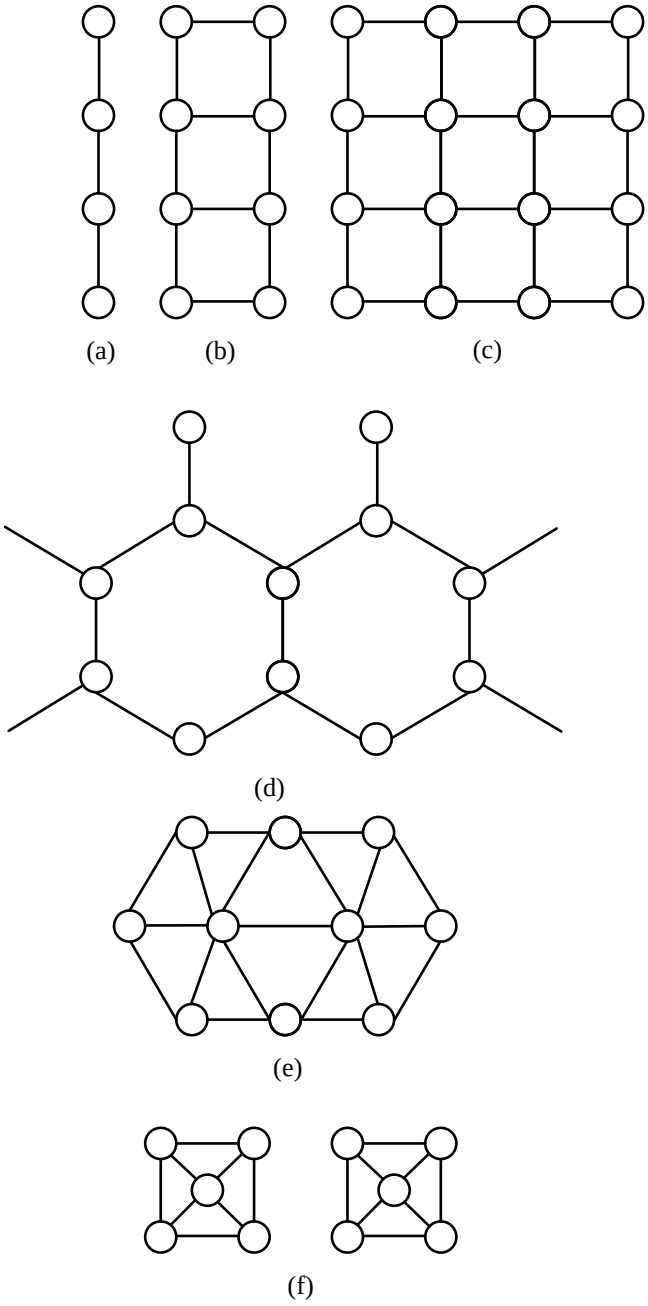


Figure 5.1.5. Examples of regular layouts. (a) 1D array, called also 1D layout, (b) 2×n 2D layout with 4 neighbors for very node, (c) Mesh 2D layout, (d) hexagonal layout with three neighbors for each node, (e) hexagonal layout with 6 neighbors for every node, (f) layout used in quantum dot cellular automata. Observe that on every non-oriented edge we can run information in one or two directions, so every edge represents one arrow (with any direction) or two arrows with opposite directions. The number of neighbors is thus a different parameter than grid connectivity, explained below.

In the DAG-based expansion (Chapter 3), all the cofactor functions that are tautological (a functional tautology of complete or incomplete functions can be used) are combined to single nodes. The nodes for functions f_a **and** $f_{\bar{a}}$ and the bus for variable *a* are mapped to the quantum layout, and the procedure is repeated for the next input variable(s) (this was illustrated in a lattice of a symmetric function in Chapter 4). The order of variables is very important, as it affects the size of the circuit and the layout area in 2D (length in 1D, volume in 3D layout).

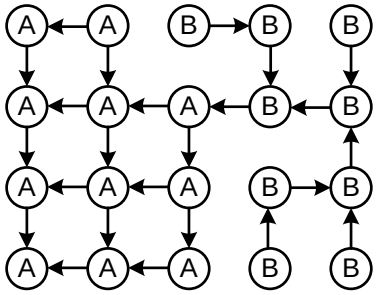


Figure 5.2.1 A Lattice (symbols A in nodes) combined with a crippled tree (symbol B in nodes). The bottom left node represents the function output. This figure demonstrates how two regular diagrams of various types can be combined in a mapping to realize a function in a 2×2 2D regular layout.

As shown in Kohavi's book [Kohavi-book] and in Aker's [Akers72] seminal paper, any single-output symmetrical binary function can in this way be directly mapped to a regular layout with 1, 2, 3, 4, ... nodes in successive levels, which correspond to input variables. The works of Kohavi have been extended by many others from over the world [Drechsler, Pierzchala, Falkowski, Sasao, Bulter] papers of Perkowski, Pierzchala, Chrzanowska-Jeske and Drechsler [Perkowski97, Perkowski97d, Drechsler94, Jeske97]. In this chapter we will show how to further extend this approach to arbitrary binary and multiple-valued functions, not necessarily only to symmetrical functions, and realize

them in Ion Trap quantum technology. I will also discuss the respective synthesis methods for incompletely specified functions (a new item, not discussed by the previous authors).

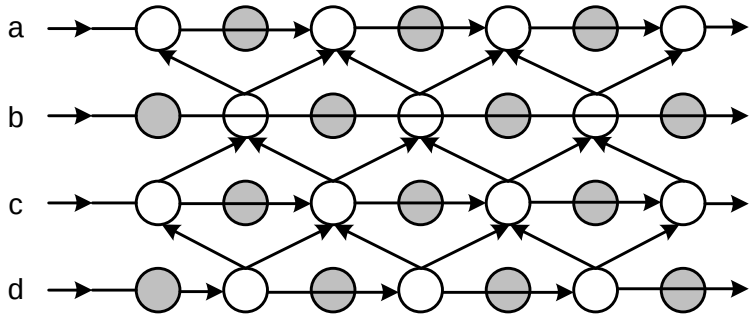


Figure 5.2.2. Another way of drawing a lattice diagram. This has to be compared for 2D Ion Trap layout with the design shown at the end of the Chapter 4. Observe the rectangular grid of dots. One half of the dots (gray ones) are used for communication only. This should be reduced. Every node has 3 predecessors and 3 successors; we thus say that a 3×3 grid is used. The neighborhood number in this example is 6. Two grid dimensions are used for the input variable bus.

As presented in this Chapter, other expansions of functions can also be used, and the expansion nodes are mapped to neighborhood structures; these regular layout structures are more powerful than those investigated theoretically in the past ([Akers72] and others). Aker's Array was designed for the worst case, so it was always large. But using the fact that small sub-functions of the functions can be realized in a regular layout as partial trees and partial lattices, much more efficient mappings to our layouts can be found. This is shown in Figure 5.2.1. In addition, the neighborhood adjacency patterns are technology-dependent, so Aker's Array or its partial lattice can also be drawn as in Figure 5.2.2 (in contrast to the ways from Chapter 4). Concluding this section:

1. Every symmetric Boolean function and many non-symmetric functions can be realized as lattices which grow linearly from level to level and are thus realizable in a 2D layout, regardless of their size.
2. Small Boolean functions can be realized as binary trees that grow less than exponentially, and can be mapped to a 2D layout.
3. Lattices and small trees can be combined with lattices on top, trees on top, or in many mixed ways and mapped to a 2D layout.
4. Thus many functions can be realized in a regular 2D layout by combining lattices and trees.
5. Other functions should be decomposed to sub-functions that can be mapped in this way.

5.2.2. Relations between logic expansions of Boolean Functions and regular diagrams.

Regular diagrams are those that can be mapped to regular layouts.

Our concept of a "regular diagram" (both irreversible and reversible) is based on four main ideas:

1. The forward expansion (distributor),
2. The reverse-expansion (collector),
3. The neighborhood geometry (structure to which expansions are mapped), like 1D, 2D or 3D quantum layouts from Chapters 2 and 4. (See Fig. 5.1.5).
4. The way to handle reversibility, by using ancilla bits, garbages, fan-outs and mirrors.

All our new design methods for combinational quantum circuits can be derived by combining the above four ideas. A node function is a Boolean function associated with a node of the diagram (node of the tree, graph, etc). A diagram is a DAG (directed acyclic graph). Nodes can have predecessor nodes and successor nodes. Function f corresponds to node that has no predecessors. Terminal nodes have no successors. Let us explain these concepts in more detail.

[1.] **Forward Expansion** of a node function creates several successor nodes of this node. Function **f** corresponds to the initial node in the diagram. The diagram is built step by step from its initial node. It is initially a tree that is transformed to a DAG in the process of its creation. Thus, a function is "decomposed", "distributed" or "expanded" to smaller "node functions". All these nodes and edges between them create the **structure** of a tree or DAG. The tree or DAG may be binary or K-ary; in general, $K \geq 2$.

[2.] **Reverse Expansion** operation merges several nodes at the bottom of the tree to create a regular diagram (before the reverse expansion, this level looks like a small, one-level forest of trees). Reverse expansion is the reverse operation to the standard expansion. As there are several standard expansions, there are also many reverse expansions. Some reverse expansions will be discussed below.

[3.] **Regular neighborhood geometry (regular layout)**, to which the nodes of the DAG are mapped, guides which nodes of a level are to be combined to keep the connections local. This geometry sets the constraint for mapping to

2D space or to 3D space. It specifies how many nodes belong to the "neighbors" of each node. In a linear array (linear layout), every node, with the exception of the first and last, has two neighbors. In a mesh architecture, every node in the middle of the geometry has 4 neighbors. A so-called 3×3 geometry is shown in Figure 5.2.2. We also call it a 3×3 "regular layout" or 3×3 "regular substrate" or "3×3 grid". Thus, we want to map a regular diagram to a regular layout.

[4.] There are several methods for making the circuit reversible. This is related to logic, not to space and placement. Do not confuse the reverse expansions with the reversible circuit that uses expansions, and with the inverse circuit. These are three independent concepts!

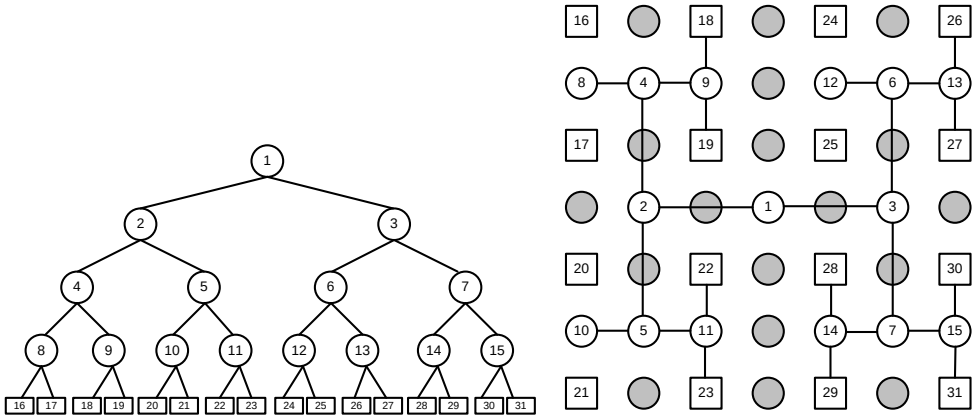


Figure 5.2.3. Binary tree (a) standard notation, (b) standard H-tree wastes spaces shown as grey circles. Nodes are enumerated for comparison.

Every signal in a regular layout can be treated as multi-valued (particularly, binary). A multi-valued (MV) connection for logic with $\mathbf{2^k}$ values can be realized by **k** binary wires

which comprise a bus (making the diagram "fat"), and which encode the multi-valued signal to binary. In general, a connection with $\mathbf{r^k}$ values can be realized by **k *r-valued*** wires which comprise a bus and encode the multi-valued signal to several multi-valued signals, if necessary. The signals in diagrams and structures shown in this chapter can be treated as multi-wire or multi-valued if it is not specifically stated otherwise. A regular diagram called a “binary tree” is shown in Figure 5.2.3. It can be realized in a “regular layout” as shown in Figure 5.2.3b. Observe that the grey circles represent ions that serve only to transmit information, so they are in a sense wasted. We want to minimize the numbers of such wasted ions. This can be done by inventing a new regular layout or by modifying the regular diagram, for instance by adding inverters or other one-argument operators between nodes.

Observe that much of the text in this chapter is independent of the radix of logic. The fact that the tree is binary does not necessarily mean that the expansions in its nodes are binary. The same is true for nodes in DAGs with three successors; they may be used in conjunction with quaternary or binary logics as well.

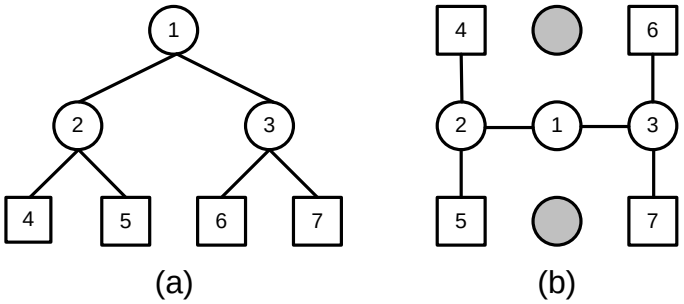


Figure 5.2.4. H-tree in logic design. (a) binary tree in standard notation, (b) H-tree, wasted space is drawn in shaded circles. Numbers corresponds to nodes.

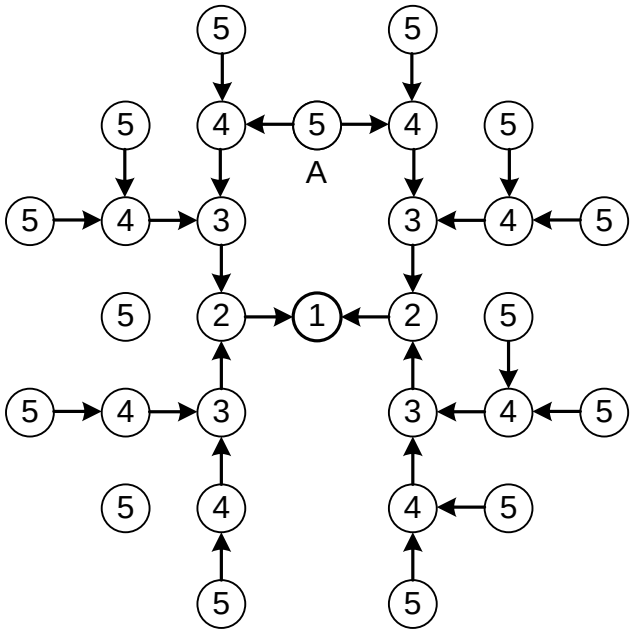


Figure 5.2.5 Illustration of constraints in drawing binary trees based on small H-trees. Numbers correspond to levels of the tree. Starting from level 5 of the mapped diagram we have no space in the regular layout 2x2 graph to insert all exponentially growing number of nodes. This is shown with node A that has two successors and is this not a tree. But we can insert these nodes if the tree is incomplete, which often takes place.

A small tree and H-tree from Figure 5.2.4, compared with the central part of the H-tree from Figure 5.2.3b, illustrate that the H-tree is not well-scalable, as the number of grey circles between nodes 1 and 2 grows from none to one. The wasted space is 2/9 = 22% in Figure 5.2.4b and 18/49 = 37% in Figure 5.2.3b. Figure 5.2.5 explains how much a binary tree can grow in 2D space. Concluding this section, many circuits can be represented by regular diagrams which are then mapped to a 2×2 regular layout. If the circuit cannot be realized as a regular layout, it can be decomposed into blocks where each block can be realized in a regular layout of some type.

5.3. Characterization of basic regular structures.

The well-known regular structures (introduced by researchers in many areas, also outside circuit design, but for instance in computer architecture and communication networks) are the following:

1. **Fat Trees**, see Figure 5.3.1.
2. **Generalized PLAs, PALs, GALs, PLAs, ROMs** see Figure 5.3.2.
3. **(Generalized) Maitra cascades, Fat Maitra cascades,** etc., see Figure 5.3.3.
4. **Akers Arrays** [Akers72], Figure 5.3.3 and **lattices**.
5. **Nets** [Perkowski95], Chapter 4.

We will show that they are all just special cases of our powerful concept of a "regular (quantum) layout".

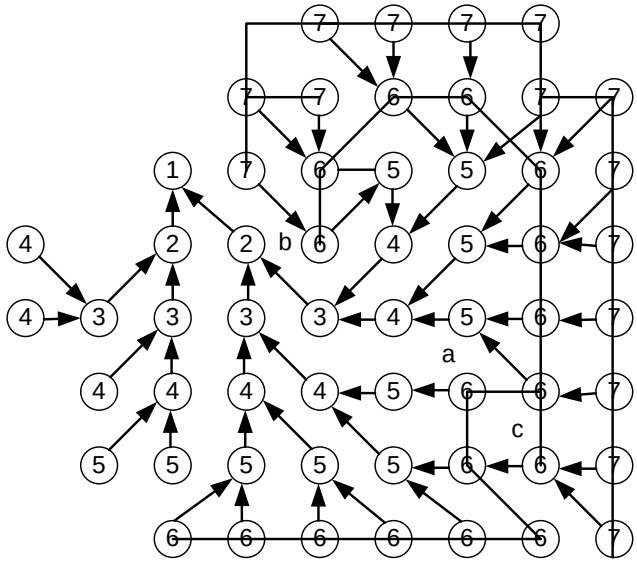


Figure 5.3.1. Regular realization of trees. An (incomplete) tree drawn on a 4×4 grid. Observe nodes a, b and c that have no space for both the predecessor nodes. The connections buses for levels 6 and 7 are also drawn. This structure shows that assuming some underlying regular substrate, certain sparse trees can be mapped to it, depending on their size. Several small trees with another neighborhood constraint (4×4 grid) are shown in Figure 5.3.1c. We call the trees that can be mapped the "crippled trees".

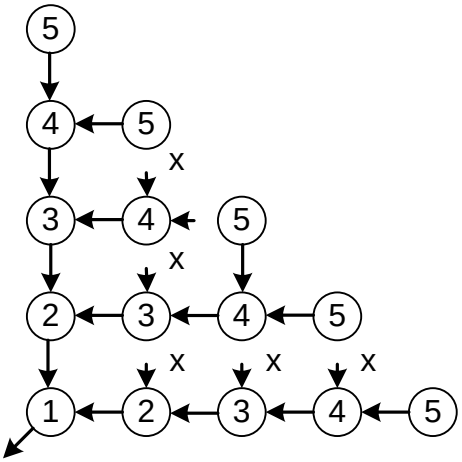


Figure 5.3.2. Crippled tree growing only to the North and East. No arrows starting from x can be completed. A 2×2 grid is used.

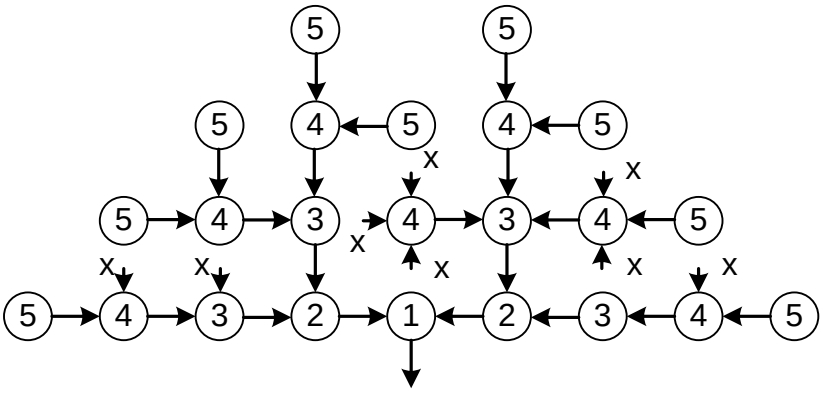


Figure 5.3.3. Crippled tree growing North, East and West. No arrows starting from x can be completed.

Figure 5.3.1 presents an example of a tree. Because the tree grows exponentially from level to level, in general a large balanced tree cannot be drawn using a 2×2, 3×3 or 8×8 grid. But because the tree structures in real-life quantum circuits are incomplete (unbalanced), a statistically high percentage of edges in them can be mapped. The reader should observe the repeated variable in level 6, and the nodes that have no space for predecessor nodes. If the lines were interpreted not as single wires but as several wires,

Figure 5.3.1 would represent a Fat Tree. The tree from Figure 5.3.1 is irregular, but is mapped to a regular 4×4 grid structure.

A crippled tree growing only in 2 directions on a 2×2 grid is shown in Figure 5.3.2. The crippled tree in Figure 5.3.3 also uses a 2×2 grid, but the tree grows in three directions. Examples of basic types of grids are given in Figure 5.3.4. Others are shown in Figure 5.3.5.

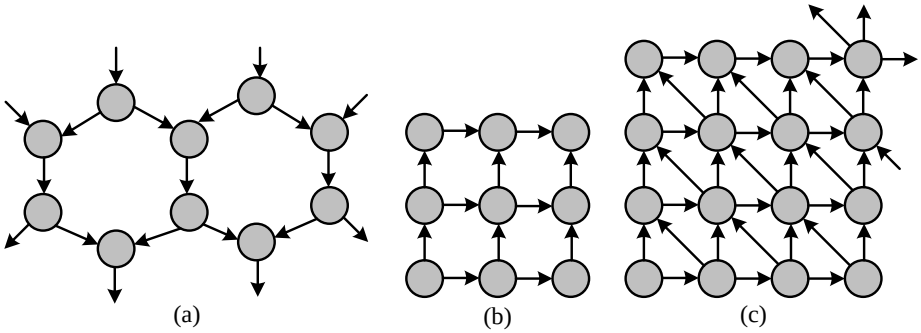


Figure 5.3.5 (a) Mixed 1×2/2×1 grid, (b) 2×2 grid, (c) 3×3 grid different than in Figure 5.3.4b.

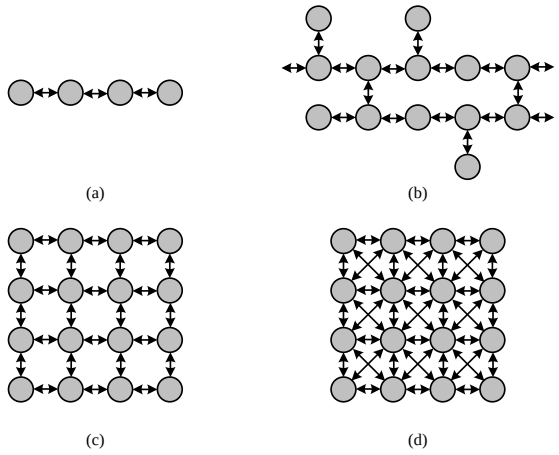


Figure 5.3.4. Basic grids, (a) 2×2, (b) 3×3, (c) 4×4, (d) 8×8.

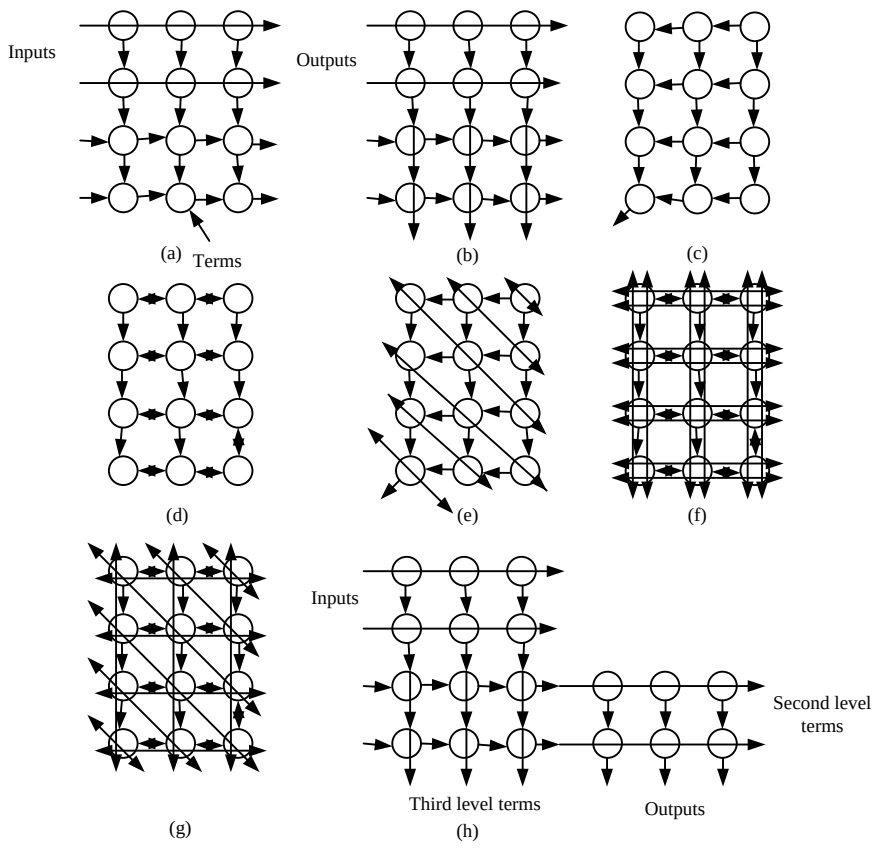


Figure 5.3.6. Various regular structures:(a) Generalized PLA of the first type, (b)Generalized PLA of the second type, (c) 2×2 pattern without input buses, (d) 3×3 pattern of FPAA and a computer network of type "mesh", note the direction of arrows, (e) 3×3 pattern of a regular diagram for binary logic with oblique buses, (f) 3×3 pattern of regular diagram with two horizontal and two vertical buses (Fine Grain FPGA architecture by company Concurrent Logic Inc.), 7×7 grid, (g) 6×6 pattern of regular diagram with horizontal, vertical and oblique buses, (h) three level PLA for TANT.

Figure 5.3.6 presents various regular structures that are either well known and taken from various sources in literature and internet, or are introduced here for the first time. Figure 5.3.6(a) presents a Generalized PLA of the first type. There is no bus through the collecting plane, only cell-by-cell abutting in it. This Generalized PLA can be a standard PLA (implementing SOP or POS logic), EXOR PLA, or an EXOR of Maitra Cascades. Maitra cascades adapted to reversible logic were presented in papers by Mishchenko and

Perkowski [Mishchenko02] and Khlopotine and Perkowski [Khlopotine02]. This structure can be mapped to a 2×2 grid. Figure 5.3.6(b) represents a Generalized PLA of the second type, with outputs from the cutting plane going as buses through the collecting plane. Figure 5.3.6 (c) is a 2×2 pattern (meaning, two inputs and two outputs) without input buses. Such pattern can be used in a systolic processor or with logic gates inside circles. Figure 5.3.6 (d) shows a 3×3 local pattern of our generalized quantum array that can be mapped to a 2D Ion Trap structure. This is also the connection structure of a computer network of type "mesh". This structure has no buses. It can be extended to a 4×4 grid by adding arrows from bottom to top between nodes. It can be also used to design regular and quantum finite state machines. In his dissertation Manjith Kumar gives examples of state machines realized with quantum memory using reversible array notation [Kumar07]. Figure 5.3.6 (e) is a 3×3 pattern of a regular diagram for binary logic with oblique buses. This will be our main pattern used to explain expansions and their layout. It is not yet certain that this is quantum realizable. An analogous pattern can be drawn for 4×4 unidirectional regular diagrams for quaternary logic, but the explanations become more complicated.

Figure 5.3.6(f) shows a 3×3 pattern of a regular diagram with horizontal and vertical buses (i.e. 7×7 grid). This corresponds directly to the (Digital) Fine Grain Field Programmable Gate Array from Concurrent Logic Inc. Figure 5.3.6 (g) illustrates a 6×6 pattern of a regular diagram with horizontal, vertical and oblique buses. This is a complete pattern of some quantum Ion Traps. Thus, pattern-wise, with the addition of one vertical and one horizontal bus, this structure would include all the previous connection structures shown above, and also many other similar structures. Some of these structures

have perhaps already been used in some designs in classical logic, but we are not familiar with these inventions. Finally, Figure 5.3.6(i) shows a three-layer PLA to realize TANT (Three-Level And-Not networks with True Inputs) networks, which is similar to a two-layer standard PLA, but has three layers of NAND gates. Similarly, multi-level generalized PLAs can be realized for an arbitrary number and type of planes. Although the examples are taken from classical logic, we believe that their quantum counterparts exist and we also want to collect as many as possible regular structures that are practical. After analyzing many structures and trying to realize some simple functions with them, we will make an informed decision which structure to use. This happens to be the array with 8 neighbors. Incidentally, this is the structure that was selected for Adiabatic Quantum Computing, but it has not been suggested yet for Ion Trap technology.

Maitra cascades are linear sequences of AND, OR and EXOR gates in any order. Fat (generalized) Maitra cascades are linear sequences of cells (iterative circuits) that have more than one carry signal between the subsequent cells. The Akers arrays look like the array from Figure 5.3.6e, with inputs repeated many times in oblique buses. These arrays are big as they are calculated always for the worst case. In addition, observe please, that our regular structure concept also extends some structures that are described in the Concurrent Logic Inc. patent application and also some other patents. In the past, I used this structure in my M.S. dissertation [DipalMS]. Concluding, all these structures can be put on 2×2, 3×3, 4×4, 5×5, 6×6, 7×7, or 8×8 grids. The highest neighborhood number is 8 (which was realized already in many VLSI/FPGA architectures, but it is an open question for Ion Trap).

In the past, the PSU team showed on many examples that this concept and geometry are very powerful and are more universal than the previously investigated general cellular structures. Here we will further extend and unify these notions to expansions **with more than 2 successor functions**, and **geometries with more than 4 neighbors.** In theory the diagram can be extended to any number of neighbors of a cell and to various $n \times k$ grids. But, based on our experience so far, we believe that in practice 8 neighbors is enough. We do not intend here to make a mathematical study of all possible structures, but only to find few useful ones in a systematic way. Observe that our structures are not always planar; some of them include non-planar circuits (like a 4 node clique), to allow realization of an arbitrary graph. But they are **always** regular. More structures can be analyzed that are also related to 2D and 3D Ion Trap technology.

5.4. The main principle of forward and inverse expansions.

In this section we will present various expansion types and we will analyze their influence on regular layouts.

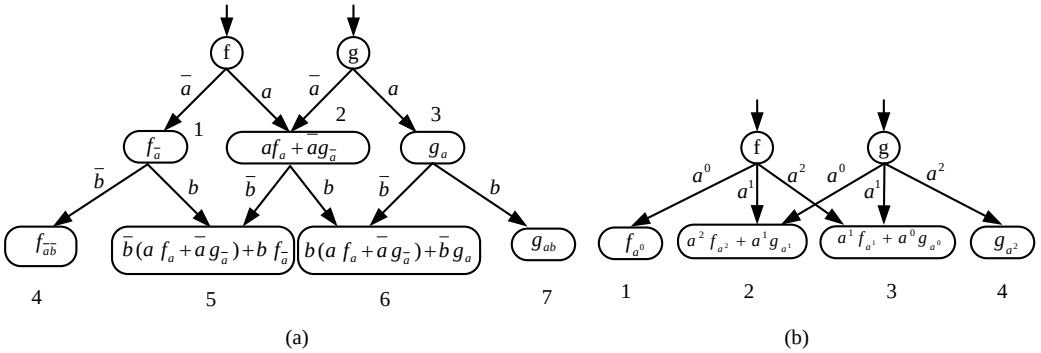


Figure 5.4.1 Forward and Reverse Expansions in 3×3 and 4×4 structures (controlling input variables in buses are not shown): (a) Shannon in binary logic, (b) Ternary Shannon-like expansion in; a generalization of binary Shannon.

Figure 5.4.1a presents the Shannon expansion with its reverse expansion. As it is easy to verify using Figure 5.4.2, a quantum array can be drawn directly from this lattice.

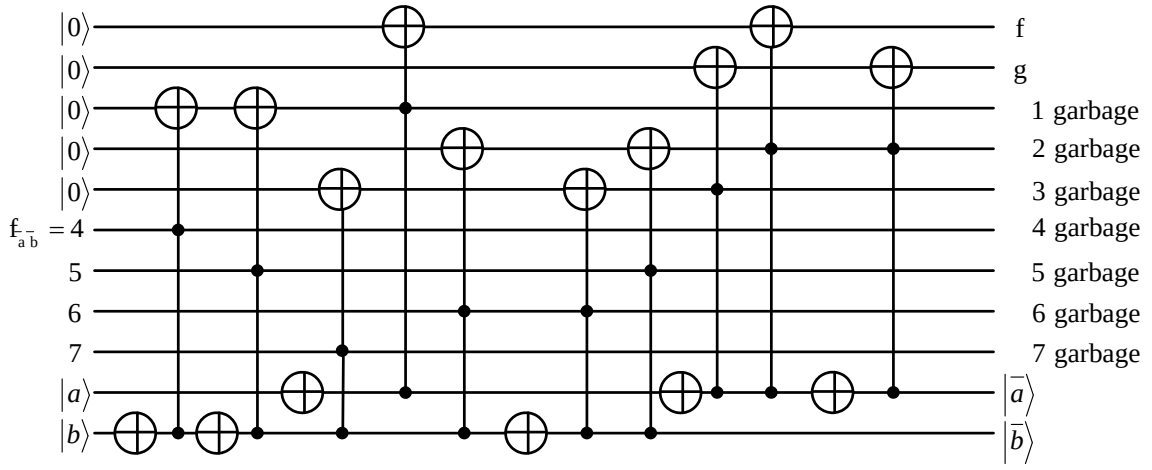


Figure 5.4.2. A Quantum Array that corresponds to forward and reverse expansions from Figure 5.4.1a is used. The only gates used are inverters and 3 × 3 Toffoli gates. Thus large k×k (k>3) Toffoli gates are avoided!

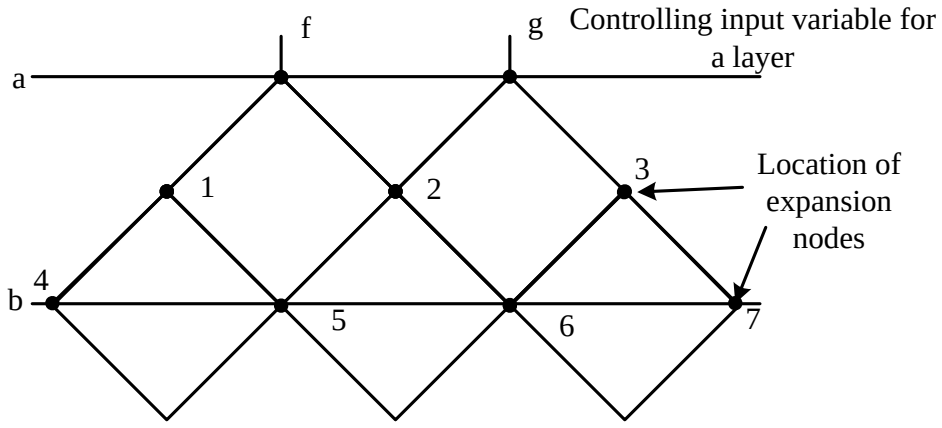


Figure 5.4.3. Symbolic schematics of quantum array structure based on 2D Ion Trap for the diagram from Figure 5.4.1a. In contrast to standard quantum array where horizontal axis corresponds to time, here both axes correspond to space and time is not shown.

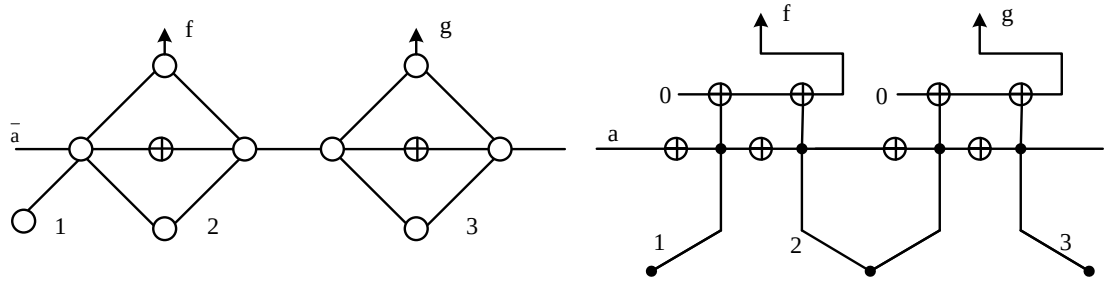


Figure 5.4.4. Notations for 2D Ion Trap realization of upper row of the diagram from Figure 5.4.1a. (a) Dots corresponds to ions, (b) vertical axis corresponds to time inside every expansion node dot. Observe black dots at the bottom representing the next level.

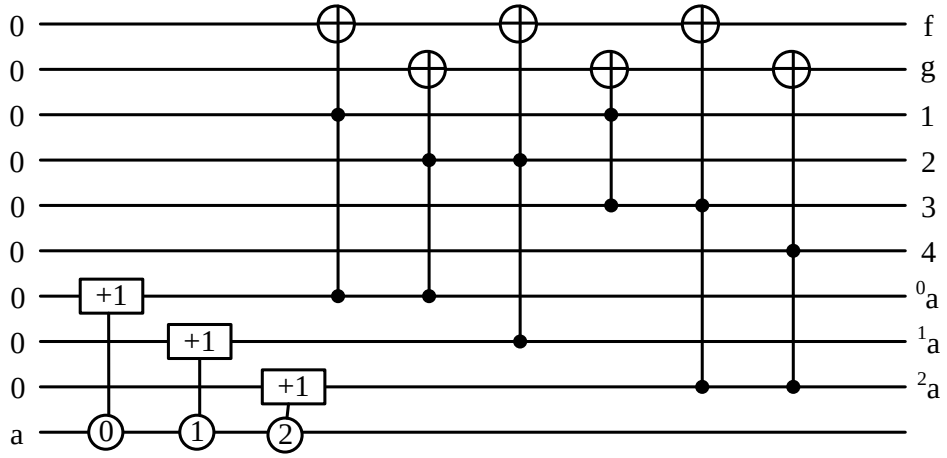


Figure 5.4.5. Ternary quantum array to realize the diagram from Figure 5.4.1b. Observe the realization of Post literals in left lower corner.

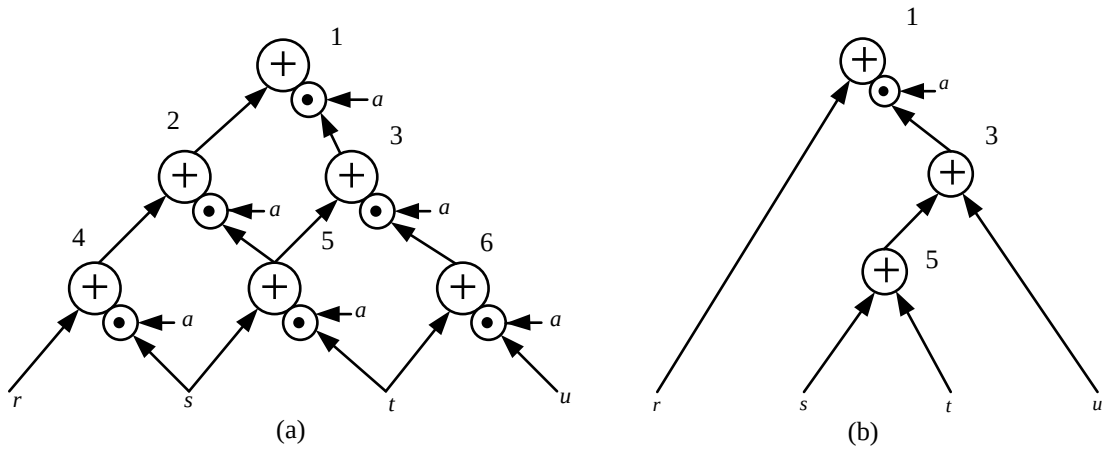


Figure 5.4.6 Explanation of trees inside regular layouts. (a) the circuit obtained directly from mapping, r = r ⊕u ⊕at ⊕at ⊕at ⊕ sa ⊕ sa ⊕sa, r = r ⊕at ⊕ au ⊕ sa, (b) the same circuit after simplifications, r = r ⊕a(s ⊕ t ⊕ u).

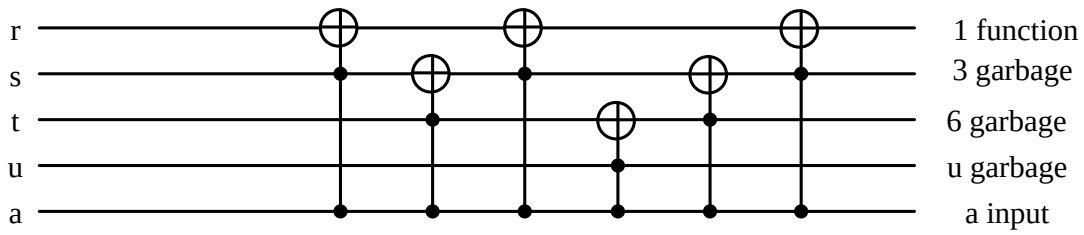


Figure 5.4.7. Quantum array for diagram from Figure 5.4.6a. The only gates used are the 3×3 Toffoli gates.

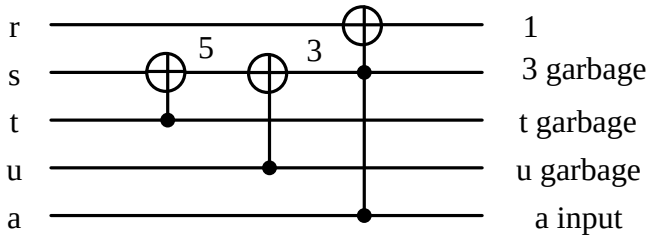


Figure 5.4.8. Binary quantum array for the simplified diagram from Figure 5.4.6b. The only gates used are the 3×3 Toffoli gates and 2×2 Feynman gates.

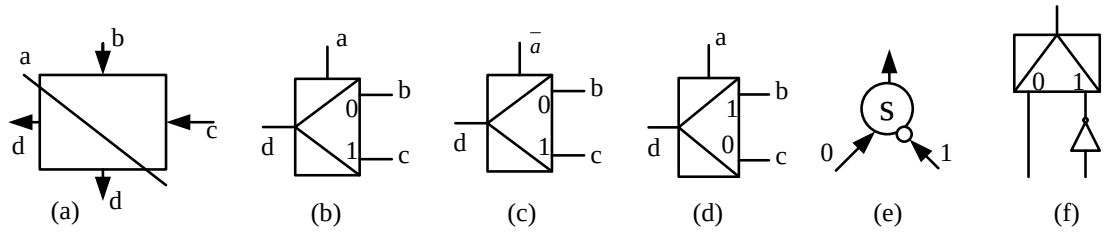


Figure 5.4.9 (a) The mux symbol in Akers Array notation, (b) standard mux symbol, (c) standard mux controlled by an inverted control variable $\bar{a}$, (d) another notation for inverted control, (e) one more notation for Shannon expansion, (f) circuit realization of Figure 5.4.9e.

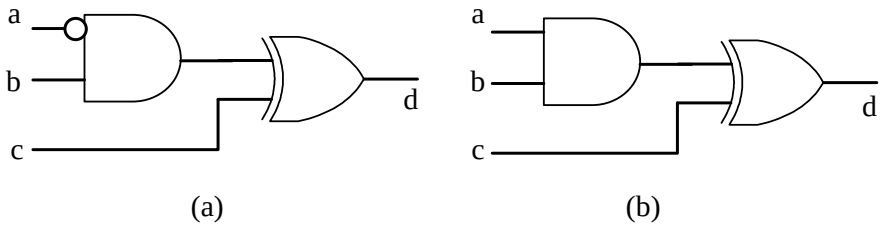


Figure 5.4.10 Davio expansions in classical notation (a) Negative Davio, (b) Positive Davio, variable a is a control variable.

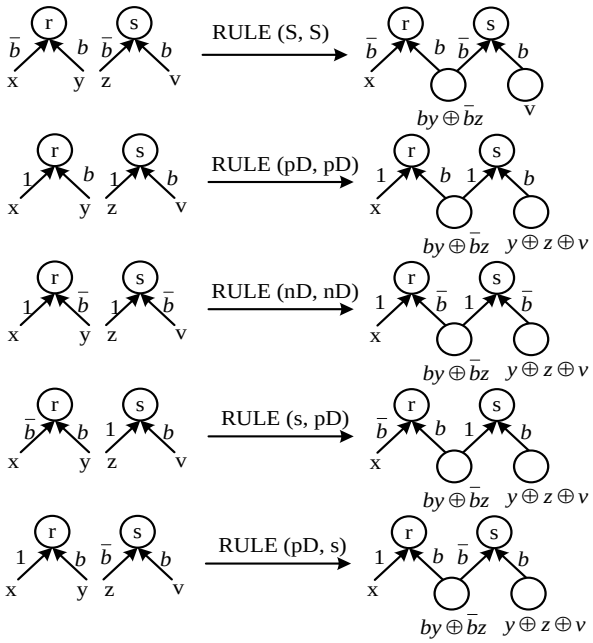


Figure 5.4.11. Expansions for Shannon, Positive Davio, Negative Davio of binary logic used in Lattices.

As discussed, Figure 5.4.11 presents two expansions, together with their inverse expansions. The expansion from Figure 5.4.11 is the familiar Shannon expansion with added reverse expansions. The expansion from this figure has its equivalent expansion for Galois Field (3) logic using Post literals. The first can be realized within a 3×3 grid. Both can be realized within a 4×4 grid. As we can see, the same principles can be used for any number of values, if the signals in the literals are disjoint. We will call this a Galois Field type of expansion. Observe that the "+ operation" in Figure 5.4.11 is OR-ing or EXOR-ing. EXOR-ing is the field operation of addition in GF(2). Observe that the "+ operation" in Figure 5.4.11 may be also maximum (MAX) in Post algebra with 3 values, thus not a field. It can also be any group operator such as GF(3) addition. Observe, however, that the same principle of collecting is used in both inverse expansions.

Remember that group operations like EXOR GF(2) or GF(3) addition do not require ancilla bits, but non-group operations like Boolean OR or MAX require ancilla bits and mirror circuits inside cells, so they complicate the design of cells.

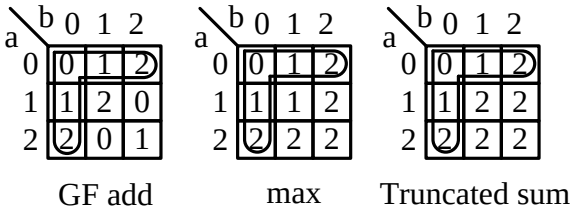
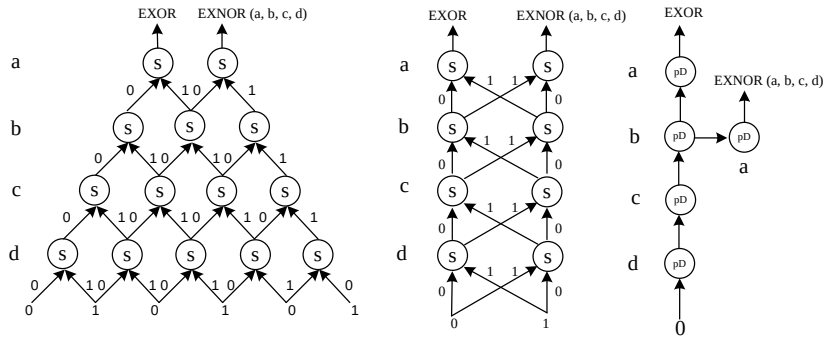


Figure 5.4.12. Comparison of addition operators in ternary algebras.

The method for creating Shannon Regular Diagrams can easily be extended to MV Shannon expansions and the associated Post Regular Diagrams for multi-output incomplete functions. In 3-valued logic, each single-variable expansion cuts a function's map to three v-cofactors, and any two of them can then be recombined by a reverse expansion operation MAX or GF(3) addition. We call this Forward and Reverse Post Expansion and Post Regular Diagram. MAX is the maximum operation denoted by +. This can be also a truncated arithmetical addition. Please observe formulas for reverse expansions of the bottom of Figures 5.4.11. All reverse expansion rules for binary logic are given in Figure 5.4.11. Let us observe that the **disjointness** of Post literals ^{0}**a,** ^{1}**a,** ^{2}**a** is the fundamental condition that must be satisfied to create maximum-type regular diagrams. It is a special case of **Orthogonality** of functions used in orthogonal expansions (which will be presented in the next section). Observe that the symbol + in Figure 5.4.6 can denote various operations, but the meaning and the method of Forward and Reverse Expansions, as well as the principle of their duality, remain the same. Comparison of ternary addition operators is given in Figure 5.4.12.

5.5. Binary Orthogonal Regular Diagrams.



(a) (b) (c)

Figure 5.5.1 Comparison of three types of Regular Diagrams for a two-output EXOR/XNOR function. (a) Shannon lattice with no internal inverters and 3×3 grid, (b) Shannon lattice with rotated nodes and 4×4 grid, (c) Positive Davio lattice and 3×3 grid (input variable buses are not shown).

Similarly, to how Functional Kronecker diagrams [Perkowski93, Shah10a] were invented by Dr. Perkowski by the generalization of the idea of BDDs, here the GF-type (MAX-type) regular diagrams from Figure 5.5.1 (b) are generalized to "orthogonal" or "Kronecker" regular diagrams. They have in general much smaller areas and a smaller number of gates. Their main advantage is seen however especially in quantum arrays for Ion Traps.

For instance, Figure 5.5.1 presents a comparison of sizes of a standard binary Shannon regular diagram and two new types of regular diagrams for the EXOR/XNOR two-output function. Figure 5.5.1 (a) presents a solution that would be obtained using the standard Shannon regular diagram or the (very inefficient in this case) basic Akers Array (that in general has repeated variables [Akers72]). The order of control variables is **a, b, c, d.** Because the function is symmetric, variables are not repeated. Observe that arrows with 0

(for negated control variable) are always on the left. The shape is a trapezoid and the size is 14 nodes. The connectivity pattern is 3×3 (input bus not included). The Akers Array [Akers72] would have (5 × 5) × 2 nodes (it realizes each of two functions separately, and uses a 5 × 5 fixed square for a 4 variable function). Figure 5.5.1 (b) presents our solution with a 3 × 3 connectivity pattern array of multiplexers. It is linear in shape and has 2 × 4 = 8 nodes. In addition to Shannon (S), the Shannon expansions with negated control variables ($\bar{s}$) are now used. Observe that arrows from the left have both 0 and 1 values. Figure 5.5.1 (c) presents a Positive Polarity Reed-Muller Regular Diagram 3×3 connectivity pattern array of positive Davio (pD) nodes. It is nearly linear in shape and has 5 nodes. This figure clearly demonstrates an advantage of having higher connection patterns and more general expansion types. Predictability and equality of delays should be a feature of all regular diagrams. All the functions in Figure 5.5.1 were symmetric, so there was no need to repeat input variables, but **what about regular diagram realization of non-symmetric functions?**

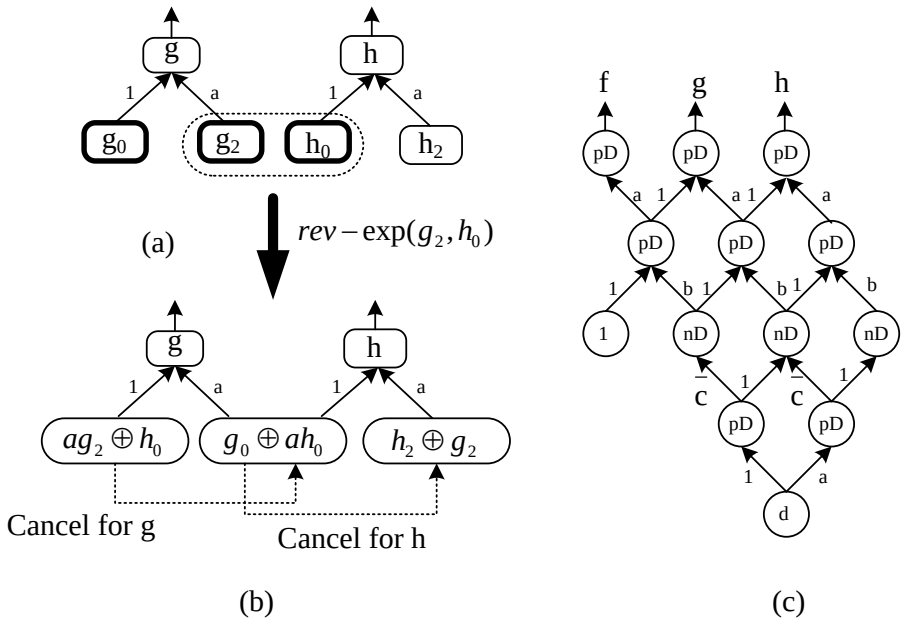


Figure 5.5.2. Creation of a Positive Davio level in a Regular Diagram: (a) two expanded nodes before reverse expansion, (b) layer of regular diagram after reverse expansion of nodes g_2 and h_0, (c) Fixed-Polarity RM Regular Diagram for functions f, g, h. Input buses are not shown for simplification.

The answer to this question is the following:

1. Reverse expressions like those in Figure 5.4.11 are used. More such expansions exist and should be investigated.
2. We can observe that more general symmetries can be considered when all types of expansions are used, not only the Shannon expansion, which is always taken into account in symmetries. These new generalized symmetries are much more general than the known symmetries of functions, so using them, more functions can be put to regular diagrams without repeating variables. These symmetries were investigated for the binary case in [Jin05, Perkowski99d, Perkowski99e].
3. Functions that do not have these more general symmetries can still be realized in regular diagrams with repeated variables.

4. Functions can be decomposed to EXORs of symmetric functions.
5. Trees and lattices can be combined, as discussed in section 5.2.

Point 3 above requires, however, use of the reverse expansions for nodes. Figure 5.5.2 a, b presents the principle of reverse expansion operation for EXOR-based, and in particular orthogonal logic. Although it is shown here only for **pD** nodes and an ordered regular diagram, the same principle is used for more complex expansions and regular diagrams of the orthogonal type. Figure 5.5.2 (a) illustrates the local situation in a level of a regular diagram after using pD expansion with respect to variable **a** to nodes **g** and **h**. In Figure 5.5.2, $g_0 = g(a=0), g_1 = g(a=1), g_2 = g_0 \oplus g_1$ etc. are the negative and positive cofactors and Boolean difference, respectively.

Figure 5.5.2 (b) presents the result of reverse expansion on the successor nodes g_2 and h_0. The reverse expanding rule is: g_2 REVERSE-EXPANSION $h_0 = ag_2 \oplus h_0$, which means that nodes representing functions $g_2 = g_0 \oplus g_1$ and h_0 are combined to a new node representing function $ag_2 \oplus h_0$. The **correction terms** ah_0 and ag_2 are propagated to the left and right, respectively. It can be easily checked that, because of the term cancelling (based on the principle $x \oplus x = 0$), in the diagram level of variable *a* from Figure 5.5.2 (b) the pD expansions of *g* and *h* are still satisfied: $g = g_0 \oplus ag_2$, and $h = h_0 \oplus ah_2$.

Figure 5.5.2 (c) presents Fixed-Polarity Reed-Muller Regular Diagram (expansions pD and nD) for functions:

$$f = a \oplus a b \bar{c} d$$

$$g = 1 \oplus b \bar{c} d \oplus a \bar{c} d \oplus a b d \oplus a b d \oplus a d$$

$$h = \bar{c} d \oplus b d \oplus a b \bar{c} d \oplus a \bar{c} d \oplus a b d \oplus a d$$

Variable *a* is used first in the pD level at the top of the diagram in Figure 5.5.2 (c). As shown in Figure 5.5.2 (c), expansions for variables *a* and *b* in the first two levels are pD, and the expansion for variable *c* in the third level is nD. Variable *a* is repeated once more in the bottom level of the diagram. The expansion in this level is pD, which means a reversed pD, that is a pD expansion with reversed role of data inputs. Observe here, that although the function is not symmetric in a standard way, it is diagram-realizable without variable repetitions because it has polarized pseudo-Kronecker symmetries. In some types of expansions the propagation of correction terms is only to the right, or only to the left. In some other expansions, especially the non-canonical ones, more powerful corrections types are created, and the algorithm selects the correction rule evaluated as the one leading to the simplest form of the next level of the diagram (fewest nodes). Selecting the order of (repeated) variables and the expansion type in each node are the most important and difficult algorithmic problems to be solved.

We can conclude that quantum circuits can be realized with regular diagrams, and that these diagrams can also be well used for binary OR-based and EXOR-based circuits and some of their MV-logic generalizations. It was already demonstrated in the dissertation

however that only some of these regular diagrams are useful for quantum circuits. In those cases where they are good, they are especially good for 2D Ion Trap Layouts. The question also arises: *"How broad are these classes of MV functions?"* Another important question is this: *"Can they be realized using our quantum array cells?"* Another question is this: *"what is the systematic structure of the space of all circuits characterized by our principles?"* These questions will be answered in Chapters 6-9.

CHAPTER 6

From Kronecker Functional Decision Diagram to Regular Quantum Array of Toffoli Gates.

In this chapter I present an efficient method for creating a quantum array from a Kronecker functional decision diagram. The quantum array created by this method always uses 3×3 Toffoli gates, Feynman gates or NOT gates, which is a unique characteristic of this method. Sufficient background on decision trees and decision diagrams was already given in the previous chapters.

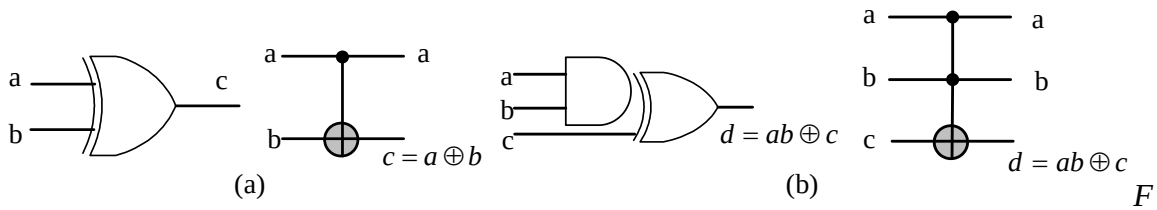
6.1. Representation of Positive Davio gate as a Toffoli gate and Invention of the d-gate.

A variety of reversible gates were presented in Chapter 2. In this subsection I present a direct relation of the 3×3 Toffoli gate as a Positive Davio gate, which is defined as follows. Many variations of the Toffoli gate can be created in a similar way.

Definition 6.1. An $n\times n$ Toffoli gate passes the first $n-1$ lines (controls) unchanged, and inverts the n^{th} line (target) if all control lines are 1.

Figure 6.1a shows representation of an EXOR gate as a reversible gate. An ancilla bit is added to repeat one of the inputs as an output, the input *a* is a control bit and input *b* is a target bit. The input *b* is inverted if $a=1$; *b* is repeated at the output if $a=0$. The EXOR gate is also known as a 2×2 Toffoli gate, a controlled-NOT (CNOT) gate, or more popularly as a Feynman gate. The Positive Davio gate is represented as a 3×3 Toffoli gate in Figure 6.1b. In this case inputs *a* and *b* are control bits and *c* is a target bit. The

control bits are repeated at the output. The target output is inverted if $a = b = 1$; c is repeated otherwise.



F igure 6.1 (a) Representation of EXOR gate as a reversible gate, Feynman Gate, (b) Representation of Positive Davio gate as a 3×3 *Toffoli gate.*

Further I describe with the help of an example the synthesis method used to create a Positive Davio Lattice in which each node represents a Positive Davio gate. Subsequently, I present an algorithm that transforms a Positive Davio lattice to a quantum array. The idea of transforming a Positive Davio lattice to a quantum array is unique and never presented in the literature. Later in this chapter I present a unique method for creating a quantum array comprised of only 3×3 Toffoli gates, Feynman gates and NOT gates. Our strong point here is that, unlike other contemporary reversible logic synthesis methods presented in [Miller03, Maslov04, Maslov04a, AgrawalJha04] that invariably use $n \times n$ Toffoli gates for *n* input reversible functions, we use only 3×3 gates (see Chapter 4 for arguments why this is more efficient).

6.2. Kronecker Functional Decision Diagram.

Decision Diagrams are efficient medium for representing logic functions. Since the early days of logic synthesis, decision diagrams have attracted many researchers working in the logic synthesis area. As we remember, an arbitrary logic function $f(x_1, x_2, x_3, ...x_n)$ can be expanded using the *positive Davio expansion, negative Davio expansion, and Shannon expansion*, as shown below [Sasao93e].

$$f = f_0 \oplus x_1 f_1 \qquad \text{(Equation 6.3)}$$

$$f = \overline{x_1} f_1 \oplus f_0 \qquad \text{(Equation 6.4)}$$

$$f = \overline{x_1} f_0 \oplus x_1 f_1 \qquad \text{(Equation 6.5)}$$

The concept of decision diagrams is very well presented as lattice diagrams in [Perkowski97]. The lattice diagrams presented in [Perkowski97] were developed from the well-known Aker's Array [Akers72], Akers's array, however, creates a large and inefficient array for all functions of *n* given variables. The next subsection provides details on Lattice Diagrams and present the method for creating Kronecker Functional Lattice Diagrams. In a subsequent stage, a method for creating a quantum array is presented.

6.3. Definition of Lattice Diagrams.

Lattice Diagrams represent data structures that describe the logic of the circuit and the regular geometry of the connections [Perkowski97]. They are thus a special case of what I call regular diagrams. Figure 6.2 shows rows and columns of an array *L*. Each entry of *L* [i, j], called a node, represents logic placed in the array. The root node of the array is *L* [1, 1]. When rotated 45 degrees clockwise each diagonal represent a level of the array, e.g., the sum for Level 1 as shown in the Figure 6.2 is 2, similarly the sum for level 2 is 3 and so on.

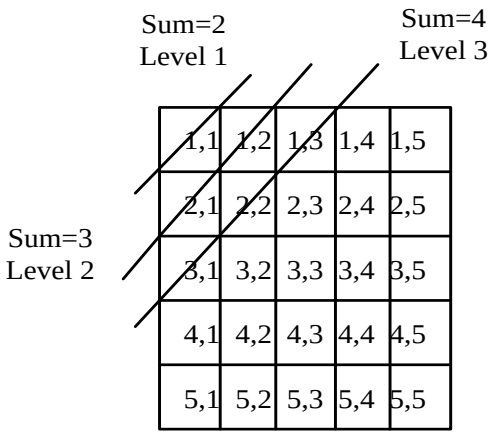


Figure 6.2. The enumeration of cells of the Akers array.

Definition 6.1. A diagonal of a matrix is a set of entries that have the same sum of indices. The sum of the indices in the first diagonal is 2, in the second diagonal is 3, and so on. A diagonal corresponds to a level of a lattice; levels are enumerated form 1. The representation of a symmetric function of *n* variables has *n* levels in the corresponding lattice.

Definition 6.2. For every entry (node) *L* [i, j] the following entries (nodes) are defined.

- The left predecessor of node *L* [i, j] is the node *L* [i, j+1]
- The right predecessor of node *L* [i, j] is the node *L* [i+1, j]
- The left successor of node *L* [i, j] is the node *L* [i, j-1]
- The right successor of node *L* [i, j] is the node *L* [i-1, j]
- The left neighbor of node *L* [i, j] is the node *L* [i+1, j-1]
- The right neighbor of node *L* [i, j] is the node *L* [i-1, j+1]

Every non-terminal node in *L* realizes a function (shown in equations 6.1 - 6.3) of its two geometric predecessors and the control variable. The geometric predecessor of a node could be a constant 0 or 1. The root node *L* [1, 1] of the array corresponds to the output of

the array. Every non-output (leaf) node of the array gives its output to one or both of its successors, hence creating connections in a regular manner to its successors in the upper level.

Definition 6.3. A *Lattice Diagram* for a single output function is represented by a Matrix *L* in which,

1. Non-zero entries *L* [i, j] correspond to logic nodes and are represented by an *expansion-type, variable, right predecessor* and *left predecessor*, where *expansion-type* is the type of expansion method presented in equation 6.1 - 6.3, *variable* is the variable used in the expansion of that node, and *right predecessor* and *left predecessor* are respective nodes or constant values.
2. A terminal node has no logical predecessor.
3. A non-terminal node has one or two logical predecessors.
4. A non-terminal node has one or two logical successors
5. For every leaf node, there exists a logic path to the output.
6. All other entries that do not represent logic nodes in the matrix have value 0 and can be eliminated from the network logic circuit.

Definition 6.4. A Lattice Diagram *realizes function F* when the function *f* obtained by its analysis forms an incomplete tautology with function *F*.

Definition 6.5. A *Functional Lattice Diagram* is an ordered lattice diagram in which all expansions are Positive Davio. It is a counter part of Positive Davio Trees and the Functional Decision Diagrams. For further details on different flavors of Lattice Diagrams please refer to [Perkowski97, Jeske97, Perkowski97a, Perkowski97b, Perkowski97c].

Definition 6.6. An *m*-input, *m*-output, completely specified Boolean function $f(X) = \{x_1, x_2, x_3 ....., x_m\}$ is reversible if it maps each input assignment to a unique output assignment.

Definition 6.7. An *n*-input, *n*-output gate is reversible if it realizes a reversible function.

6.4. Creating a Positive Davio lattice.

An example of creating a Positive Davio lattice diagram for a single output function is shown in Figure 6.3. Positive Davio expansions are used in each node and pD. A pD joining operation as shown in Figure 5.4.11 is used. Positive Polarity Reed-Muller forms are used in nodes to represent the function. For example for the function shown below, a Positive Davio lattice is created as follows.

$$f = 1 \oplus ad \oplus bd \oplus abd \oplus ac \oplus bc \oplus cd \oplus bcd$$

Variable *c* is selected for expansion in the first level, and second level nodes are created as a result of this expansion.

$f_{\bar{c}} = 1 \oplus ad \oplus bd \oplus abd$, which is the left node of the second level.

$f_c = 1 \oplus ad \oplus abd \oplus a \oplus b \oplus d$

$f_c \oplus f_{\bar{c}} = a \oplus b \oplus d \oplus bd$, which is the right node on the second level.

The variable *d* is selected for the second level. The right co-factor of the node $f_{\bar{c}} = 1 \oplus ad \oplus bd \oplus abd$ is $a \oplus b \oplus ab$. The left co-factor of the node $a \oplus b \oplus d \oplus bd$ with respect to variable *d* is $a \oplus b$. The joining operation on this co-factor is computed as follows.

$$d(a \oplus b \oplus ab) \oplus \bar{d}(a \oplus b)$$
$$= da \oplus db \oplus dab \oplus \bar{d}a \oplus \bar{d}b$$
$$= a \oplus b \oplus abd$$

which is a left node on the third level. The diagram is completed in a similar fashion. The final diagram is shown in Figure 6.3.

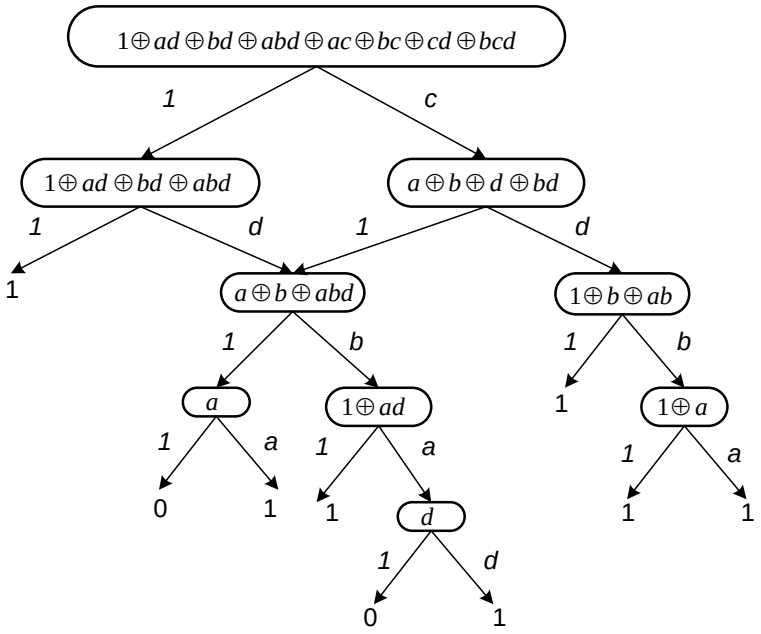


Figure 6.3 Positive Davio lattice for an example function.

6.4.1. Algorithm pseudo code for creating Lattice Diagrams.

The algorithm for creating Lattice Diagrams is described in this subsection. This algorithm can be used for creating different types of Lattice Diagrams such as Positive Davio, Shannon, and Negative Davio. However, minor changes in the operations on the Boolean function generated for each node are needed. For future development I will advance my software to provide the user a menu selection to choose amongst different Lattices. The input to the algorithm is a Boolean function and the order of variables which will be used in expanding nodes at every level. The Boolean function is the output

of the root node of the Lattice. A Quantum Array is created for the lattice diagram created using this algorithm. The algorithm to create a Quantum Array from the lattice diagram is described in the next section.

```
Input Boolean function;
Input Order of variables for a given Boolean function;

level = 1;
node_number = 1;
root_node = Input Boolean function;
variable = current_variable;
Add root_node as a first element of a linear linked list;

Create PCF (Positive cofactor) for a root_node using variable;
PCF.level = level;
PCF.node_number = 2×node_number;

Create NCF (Negative cofactor) for a root_node using variable;
NCF.level = level;
NCF.node_number = (2×node_number) + 1;

Simplify PCF and NCF to eliminate common terms;
Add PCF and NCF of the root_node to the linear linked list (at the end of a
linked list);

while (while all nodes in a Lattice are !constant)
{
  variable = Next variable in the list;
  ++level;
  foreach (node in the level)
  {
    Create PCF using variable;
    PCF.level = level;
    PCF.node_number = 2×(node_number of the parent node);

    Create NCF using variable;
    NCF.level = level;
    NCF.node_number = (2×(node_number of the parent node)) + 1;

    Simplyfy PCF and NCF to eliminate common terms;
    Add PCF and NCF to the linear linked list;
```

```
            Traverse the linked list to go to the level in current iteration;
            Identify nodes eligible for performing joining operation in the current
level;
            Perform joining operations;
            Simplify the function of this new node to eliminate common terms;
            Delete nodes on which joining operation was performed;
            Add new node generated with node number;
          }//end Foreach
        }//end while, Lattice diagram is complete

//Display the Lattice diagram

        while (!end of the linked list)
        {
          Display Boolean expression in the node;
          Display variable used by the node;
          Display level;
          Display node_number;
      }//end while
```

Figure 6.4 Algorithm for implementation of the Lattice Diagrams.

6.5. Description of the Algorithm (KFDD to QA with Toffoli gates).

This section describes one of the main ideas of this dissertation, which is creating a Quantum Array from Kronecker Functional Lattice Diagram (KFDD). We also discuss the implementation of this algorithm with the help of an example. It is assumed that a given function in Positive Polarity Reed-Muller (canonical) form is synthesized into a Positive Davio lattice (KFDD). Inputs to the algorithm are a synthesized KFDD and the functional output (root node) node of the decision diagram. The output of the algorithm is a Quantum Array that consists of a cascade of 3×3 Toffoli gates, Feynman gates or NOT gates. The Quantum Array is created by forming layers of cascades of these gates. Each gate in the array is either a 3-qubit Toffoli gate, a 2-qubit Feynman gate or a 1-qubit NOT gate, which is a unique characteristic of our method. It is easy to understand that the positive Davio cell used in creating a KFDD can be presented as a Toffoli gate as shown

in Figure 6.1. Each node in the KFDD represents a Toffoli gate. All the gates that we use are of low cost and are optimized once for all. The only price that we pay is adding more than the minimum number of ancilla bits, but this is not a big problem for some quantum technologies.

The algorithm performs preorder traversal of the Kronecker Functional Decision Diagram to find each output of the quantum array. For each output node further preorder traversal is performed to create a layer of the quantum array. This method creates a cascade of reversible gates connected in subsequent stages, thus creating a layer that consists of cascades of the reversible gates. Each Toffoli gate in the array receives one input from the variable that was used in the expansion in order to create the same node and the other input from the output of the gate one layer above. This rule is invariably true for all Toffoli gates in the array created using this method. Each Feynman gate receives input from the variable used for expansion of the same node. The pseudo-code of the algorithm is presented in Figure 6.5.

```
Algorithm 1. Create Quantum Array from the Kronecker Functional
Lattice Diagram
Input: 1. Functional Kronecker Lattice   Diagram
       2. Root node of a given KFDD
i ← 1;
j ← 1;
root_node ← [i, j]
//Traverse all nodes of the Lattice using preorder traversal to identify the
output nodes
node ← root_node
enqueue (Q, node)
preorder (node) {
                    if (node.RS == NULL)
                    enqueue (Q, node)
                    if (node.LP != NULL) then preorder (node.LP)
                    if (node.RP != NULL) then preorder (node.RP)
```

```
            }

Foreach (node in Q)  {
                        while (node.LP !constant) {
                        determine_gate (node)
                        enqueue (Q1, node)
                        node  ← node.LP
                                    }
                        if (node.LP == constant)
                        determine_gate (node)
                        enqueue (Q1, node)

                        dequeue Q1
                    }
determine_gate ( node) {
                        if (node.RP == 0)
                        node ← WIRE
                        else if (node.RP == 1)
                        node ← FEYNMAN
                        else
                        node ← TOFFOLI
                        return
                    }
```

Figure 6.5. pseudo-code for KFDD to QA algorithm.

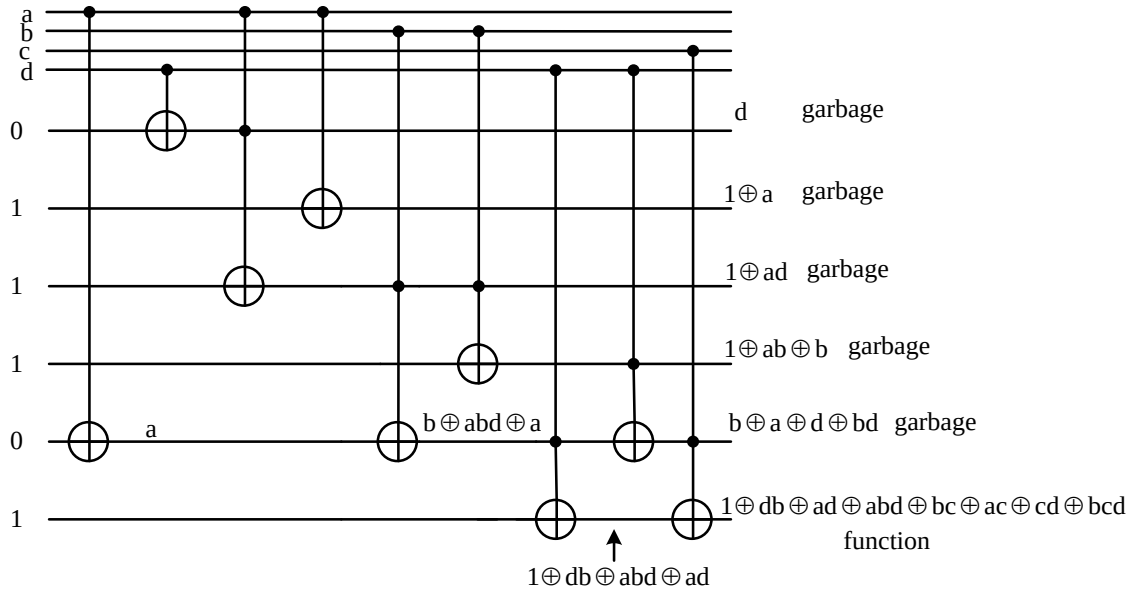


Figure 6.6. Quantum array for the Positive Davio Lattice from Figure 3.

Figure 6.5 describes the steps of the algorithm. I present an example for the detailed explanation of our method. Consider a KFDD shown in the Figure 6.4, which implements the function:

$$f = 1 \oplus ad \oplus bd \oplus abd \oplus ac \oplus bc \oplus cd \oplus bcd$$

The functional output *f* is a root node. For convenience all nodes of KFDD are labeled as shown in Figure 6.7. First we identify the output nodes of the array by preorder traversal of the KFDD and create a queue of the output nodes. The first node in the queue is the functional output (root node) node N[1, 1]. If the left predecessor (LP) of the node is constant, the algorithm stops traversal of the branch. In our example the left predecessor of N[1, 1] is node N[2, 1]. The left predecessor of node N[2, 1] is a constant 1, so traversal will stop and the node N[1, 1] is added in the queue Q. In the next step node N[2, 2] will be explored using preorder traversal for the right node of the node N[2, 1]. The left predecessor of this node is node N[3, 2], which terminates with the constant values. The right predecessor of the node N[2, 2] is node N[2, 3]. This node doesn't have a right successor and hence node N[2, 3] is considered as a (garbage) output of the quantum array and will be added to the queue Q. All nodes of the KFDD are explored in a similar fashion to complete the queue. The queue contains an ordered list of functional output nodes of the array. At the end of this step we will have nodes N[1, 1], N[2, 3], N[2, 4], N[1, 2], N[1, 3], N[1, 4]. The nodes that are explored more than once and qualify as output nodes will not be added to the Q. Only the first entry of the node will be registered in the queue.

In the second step we create a layer of cascades of reversible gates by traversing left for every node in the queue. We also create a new queue for every output node created in the previous step. In our example node N[1, 1] will be selected first. The left predecessor of this node is N[2, 1] and the right predecessor is N[1, 2]. The procedure determined gate will be used for N[1, 1] which will result in a TOFFOLI gate as the right predecessor is not a constant value. The output of N[2, 1] acts as a target input of the Toffoli gate represented by N[1, 1]. The output signal from N[1, 2] acts as one of the control inputs, as does the variable *c* which is used in the expansion of the root node N[1, 1]. This is shown as the output node in the bottom layer of Figure 6.6. Next N[2, 1] will be explored; as the left predecessor of this node is the constant 1, no further node needs to be explored by traversing left, and constant 1 will be the target input for the Toffoli gate represented by N[2, 1]. The two control inputs of N[2, 1] are the variable *d* and the output signal of the node N[2, 2]. The node N[2, 1] represents the second gate in the bottom layer of Figure 6.6. This will complete the bottom layer of the quantum array represented by the KFDD of Figure 6.3. Other layers of the cascade of Toffoli gates are completed in a similar fashion in order to complete the array. The final Quantum Array is shown in Figure 6.6. For more elaboration another example of a six variable symmetric function is shown in Figure 6.8.

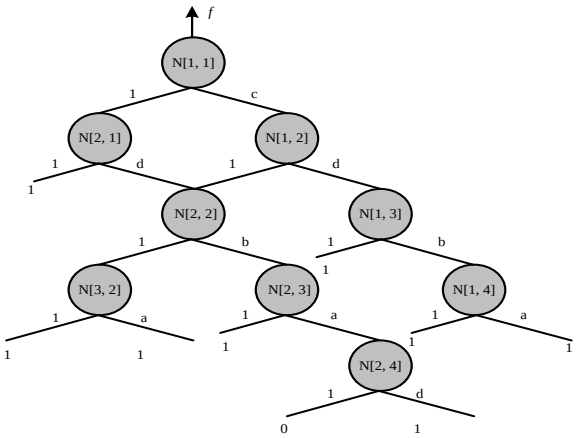


Figure 6.7. Representation of Positive Davio Lattice.

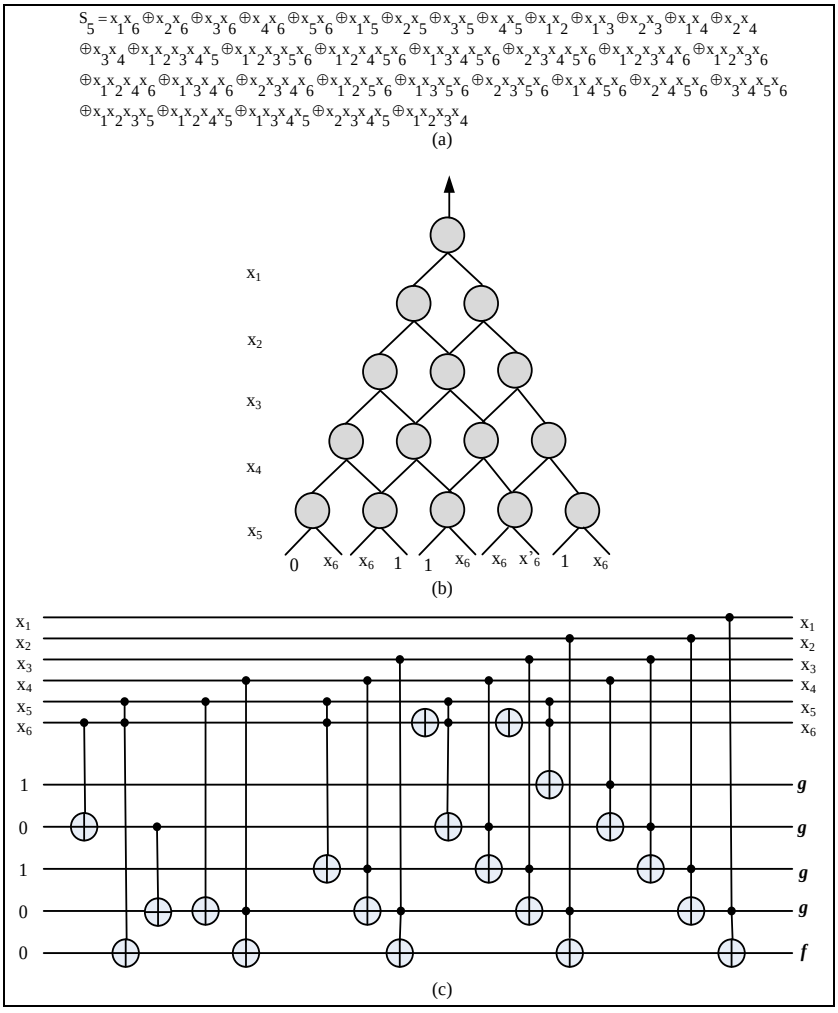


Figure 6.8. (a) Six variable symmetric function (b) KFDD created for six variable symmetric function (c) quantum array created for the six variable symmetric function.

6.6. Quantum Array Optimization by Creating Efficient KFDD.

The KFDDs are directed acyclic graphs (DAG) representing the logic of a circuit and the geometry of its connections. The data structure created by a KFDD is similar to the popular ordered binary decision diagram (OBDD) and the pseudo-symmetry Binary Decision Diagram (PSBDD) presented in [Jeske97]. It is evident that the quantum cost of the array created for a given specification using our algorithm depends on the number of levels and the structure of the KFDD created for the function. For symmetric functions variable repetition is not required and the number of levels in the KFDD will be same for any order of variables. Also number of levels will be same as number of variables used in the function. However in case of the non-symmetric functions, the joining operation performed to merge neighboring non-isomorphic nodes in the KFDD introduces new variables (that are used for expansion in the previous stage) in the logic functions represented by the merged nodes. This enforces repetition of variables in the subsequent stages. It is hence inevitable to use appropriate ordering of variables to minimize the size of the KFDD.

Different approaches for variable ordering are presented in [Rudell93, Panda94] that can be readily used for KFDDs. The adjacent isomorphic nodes can be presented with a single node and joining operation is not required. To minimize repetition of variables in KFDDs, selection of variables has to be such that geometric adjacencies are preserved. To provide further freedom to increase probability of adjacencies for isomorphic nodes, a flip Davio operation as shown in Figure 6.9 is performed similar to the flip Shannon operation for PSBDD presented in [Wang01].

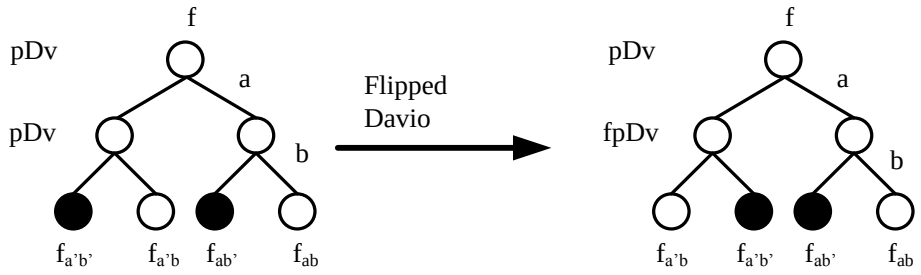


Figure 6.9. Flipped Positive Davio Node.

For any pair of variables x_i and x_j there are four cofactors, fx_ix_j, fx'_ix_j, $fx_ix'_j$, $fx'_ix'_j$. The function is symmetric in these two variables if any two of the four cofactors are equivalent. Negation of any one of the variable can also be used, symmetry created by negation of any one variable is called skewed-symmetry. Six possible symmetries are presented in [Wang01] is shown in Figure 6.10 for better understanding. We used these symmetry rules in creating optimum KFDD and to minimize the quantum cost of the array.

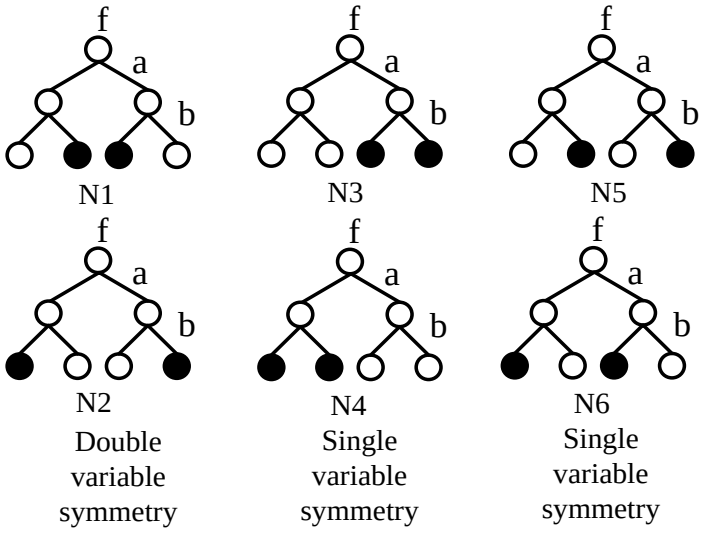


Figure 6.10. Representation of six symmetries.

The window permutation algorithm explained in [Rudell93] proceeds by selecting a level *i* in the KFDD and exhaustively searching all *k*! permutations of the *k* adjacent variables starting at level *i*. This is done by selecting *k*!-1 pair wise exchanges followed by up to *k*(*k*!-1)/2 pair wise exchanges to restore the best permutation obtained during the process. This process is then repeated for each level in the KFDD. The Figure 6.11 shows the variable permutations that are explored when applying a window of size $k = 3$ starting at variable *b*. Total five permutations are explored with four adjacent variable swaps, then three additional variable swaps are used to restore the best permutation. The cost of the KFDD is recorded when variable exchange is done at the given level. The window permutation algorithm is practical for functions up to five variables.

a, b, c, d, e	Initial
a, c, b, d, e	swap (b, c)
a, c, d, b, e	swap (b, d)
a, d, c, b, e	swap (d, c)
a, d, b, c, e	swap (c, b)
a, b, d, c, e	swap (d, b)
a, b, c, d, e	swap (b, d)
a, c, b, d, e	swap (b, c)
a, c, d, b, e	swap (b, d)

Figure 6.11. Example of a window permutation algorithm.

The Sifting algorithm presented in [Rudell93] for OBDD minimization can be effectively applied for KFDD minimization. This algorithm is based on finding the optimum position for a variable assuming all other variable in a fixed position. The Sifting algorithm works as follows. The variable is exchanged with its successor variable until the variable becomes the next to last variable in the KFDD, this means the variable is shifted to the

bottom of the KFDD. Then the variable is exchanged (backwards) with its predecessor until the variable is at the top of the KFDD. The best size of the KFDD is stored during the process and variable is finally placed at its optimum position. The example of Sifting algorithm is shown in Figure 6.12.

a, b, c, d, e	Initial
a, c, b, d, e	swap (b, c)
a, c, d, b, e	swap (b, d)
a, c, d, e, b	swap (b, e)
a, c, d, b, e	swap (e, b)
a, c, b, d, e	swap (d, b)
a, b, c, d, e	swap (c, b)
b, a, c, d, e	swap (a, b)
a, b, c, d, e	swap (b, a)
a, c, b, d, e	swap (b, c)
a, c, d, b, e	swap (b, d)
a, c, d, e, b	swap (b, e)

Figure 6.12. Example of Sifting algorithm..

6.7. Experimental Results.

The Table 6.1 shows experimental results in terms of number of gates in the synthesized circuits as well as respective quantum cost. The quantum costs of the reversible gates are computed as method used in contemporary quantum CAD algorithms [Maslov04]. The Table 6.1 also shows comparison of results with other contemporary algorithms. The best quantum cost results are marked as italic and bold. Dashes in the Table mean no results are available or possible. It is evident that our algorithm generated better results in terms of quantum cost of the circuit for most of the benchmark functions. Our algorithm is particularly efficient for functions with large number of variables as it invariably synthesize reversible circuits with 3×3 Toffoli gates, Feynman gates and NOT gates. The

synthesis runs were done on a MAC machine with Intel 2GB Core2 Duo processor with 2.4GHZ, synthesis run time for each benchmark function is shown in column 6.

TABLE 6.1

Benchmark	#Real inputs	#Garbage inputs	#Gates Lattice	Cost Lattice	CPU time Lattice	#Gates DMM	Cost DMM	#Gates AJ	Cost AJ
2to5	5	4	31	107	0.12	15	107	20	***100***
rd32	3	1	4	***8***	< 0.01	4	***8***	4	***8***
rd53	5	5	11	***39***	< 0.01	16	75	13	116
rd84	8	7	20	***68***	<0.01	28	98	-------	------
5bitadder	10	5	29	***55***	<0.01	***29***	***55***	-------	------
8bitadder	16	8	122	***322***	0.10	***122***	***322***	-------	------
3_17	3	1	10	21	< 0.01	6	***12***	6	14
6sym	11	4	19	75	0.37	20	***62***	NA	NA
9sym	15	5	25	101	0.40	28	***94***	NA	NA
5mod5	5	1	14	***58***	< 0.01	10	90	11	91
4mod5	4	1	6	18	< 0.01	5	***13***	5	***13***
ham3	3	0	3	***7***	< 0.01	5	***7***	5	9
ham7	7	4	21	61	<0.01	25	***49***	23	81
ham15	15	9	47	***191***	0.10	109	206	-------	------
xor5	5	0	4	***4***	< 0.01	4	***4***	4	***4***
xor20	20	0	19	***20***	<0.01	19	***19***	19	***19***
xnor5	5	1	5	***5***	< 0.01	------	-------	--------	-----
decod24	4	2	10	***30***	< 0.01	------	-------	11	31
Cycle10_2	12	6	180	***860***	27.9	19	1198	-------	------
Cycle17_3	20	10	920	***4160***	40.1	48	6057	-------	------
ham7	7	5	22	***58***	0.10	23	81	24	68
Graycode6	6	5	5	***5***	< 0.01	5	***5***	5	***5***
Graycode10	10	9	9	***9***	< 0.01	9	***9***	9	***9***
Graycode20	20	19	19	***19***	< 0.01	19	***19***	19	***19***
nth_prime3_inc	3	4	4	***6***	< 0.01	4	***6***	-------	------
nth_prime4_inc	4	5	16	***48***	< 0.01	12	58	-------	------
nth_prime5_inc	5	5	29	91	0.22	26	***78***	--------	------
nth_prime6_inc	6	6	148	***586***	0.36	55	667	-------	------
Alu	5	2	5	17	< 0.01	------	-------	18	114
4_49	4	4	16	***52***	0.04	16	58	13	61
hwb4	4	4	12	***28***	< 0.01	17	63	15	35
hwb5	5	5	24	***96***	1.2	24	104	-------	------
hwb6	6	6	32	***128***	2.0	42	140	-------	-----
hwb7	7	6	49	***185***	0.10	35	203	-------	-----
pprm1	4	4	9	***33***	< 0.01	------	------	-------	-----
pprm2	10	6	55	***235***	0.50	------	------	--------	-----
pprm3	15	12	29	***540***	0.50	------	------	--------	-----

CHAPTER 7

Development of the Dipal Gate Family and Design of the Layered Diagrams.

7.1. Layered-based Regular Structures with Shannon Expansion.

The Shannon Lattice can be built for incompletely specified functions. This method can be next generalized to other regular diagrams. I assume here that each binary function *f* is represented by a pair [ON(f), OFF(f)]. Thus all cofactors f_a for the product of literals a, are pairs: f_a = [ON(f_a), OFF(f_a)]. The Figure 7.1a explains the principle of creating a Shannon Regular Diagram based on ordered Shannon expansions for a multi-output function. The direction of arrows shows the expansion flow. Observe that every cofactor f_a of the product a of an (in)complete function f can be interpreted as intersecting f with a and replacing all K-map cells outside product **a** with don't cares. A standard cofactor $\mathbf{f_x}$ where **x** is a variable does not depend on this variable. In my interpretation, though, $\mathbf{f_x}$ is still a function of all variables including x, but as a result of cofactoring the variable **x** becomes vacuous. This is called a **vacuous cofactor**, and denote by **v-cofactor**.

Thus, for any two disjoint products a_1 and a_2, the v-cofactors f_{a1} and g_{a2} are disjoint (observe that standard cofactors are in general not disjoint), therefore functions f_{a1} and g_{a2} are in an incomplete tautology relation, and functions f and g are not changed when f_{a1} and g_{a2} are combined (OR-ed) to create a new function: $a_1 f_{a_1} + \bar{a}_2 g_{a_2}$ as in Figure 7.1a (where: $a_1 = a_2 = a$, and $\bar{a}$ is denoted as a'. This is called the Reverse Shannon Expansion. (Observe that for every expansion one can create a reverse expansion).

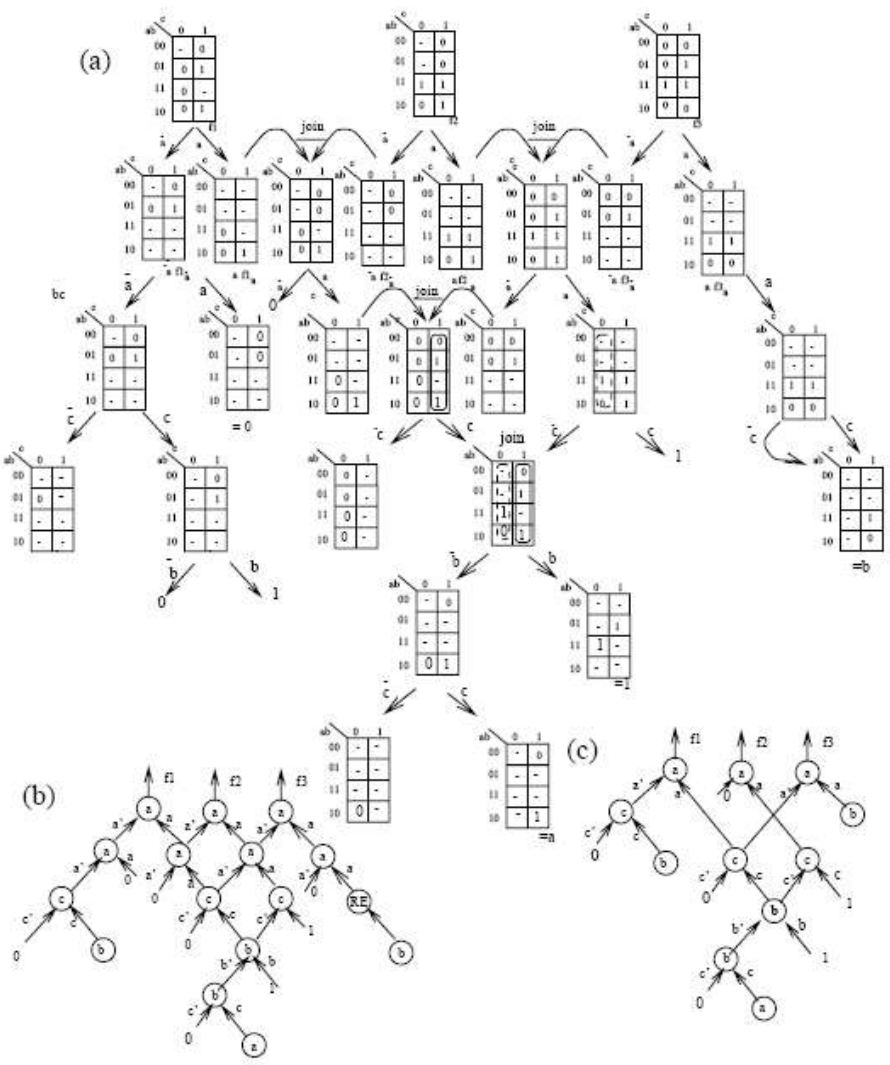


Figure 7.1 Shannon Expansions and Reverse Expansions for Incomplete multi-output binary function.

This way, the entire layout is created level-by-level, only three levels shown in Figure 7.1a. Observe that functions in the regular layout nodes become more and more unspecified when variables in levels are repeated, and ultimately nodes become constants, which terminates the layout generation process. This way, every variable cuts a Kmap into two disjoint parts, arbitrary two functions f and g can always be expanded together to a Shannon diagram, with OR-ing as a reverse expansion operation, provided

that the same variable x_i is used in the level, and all expansions use negated literal $\overline{x_i}$ in the left, and positive literal x_i of the variable in the right.

As shown in Figure 7.1, arbitrary functions of the same arguments are cut in half by expansion of each variable and new functions in levels are created by rearranging the cofactors in reverse expansions. This process can lead to a slight increase of the number of nodes in comparison with a **shared OBDD** of these functions. But a regular structure is created, thus simplifying **quantum layout** and making delays predictable. In case when the products a_1 and a_2 are **not** disjoint, the v-cofactors f_{a1} and g_{a2} can, in some cases, still form an incomplete tautology of functions. When these two cofactors satisfy a tautology relation, then functions f_{a1} and g_{a2} can be combined (OR-ed) without changing functions f and g. Obviously, the same method works for arbitrary number of output functions. Figure 7.2a shows example of binary Shannon Lattice and Figure 7.2b shows its realization in a quantum array. The regularity here is limited because it is not scalable. This is why we realized this lattice in 2D Ion Trap layout in Chapter 4 (at the end). Although this design is less efficient in terms of added ancilla bits than the method from section 6.5, still the fan-out can be realized without additional ancilla bits. Moreover, this circuit can be prepared for cascading oracles by adding mirror gates of all gates other than that realizing the output mux. This way the constants are reinitialized in ancilla bits. This way, classical Shannon Lattices can be also mapped to 1D and 2D quantum layouts to be realized in Ion Traps. The above presented synthesis method is very general and is applied to any gates from the newly developed "Dipal Gate" family later in this chapter.

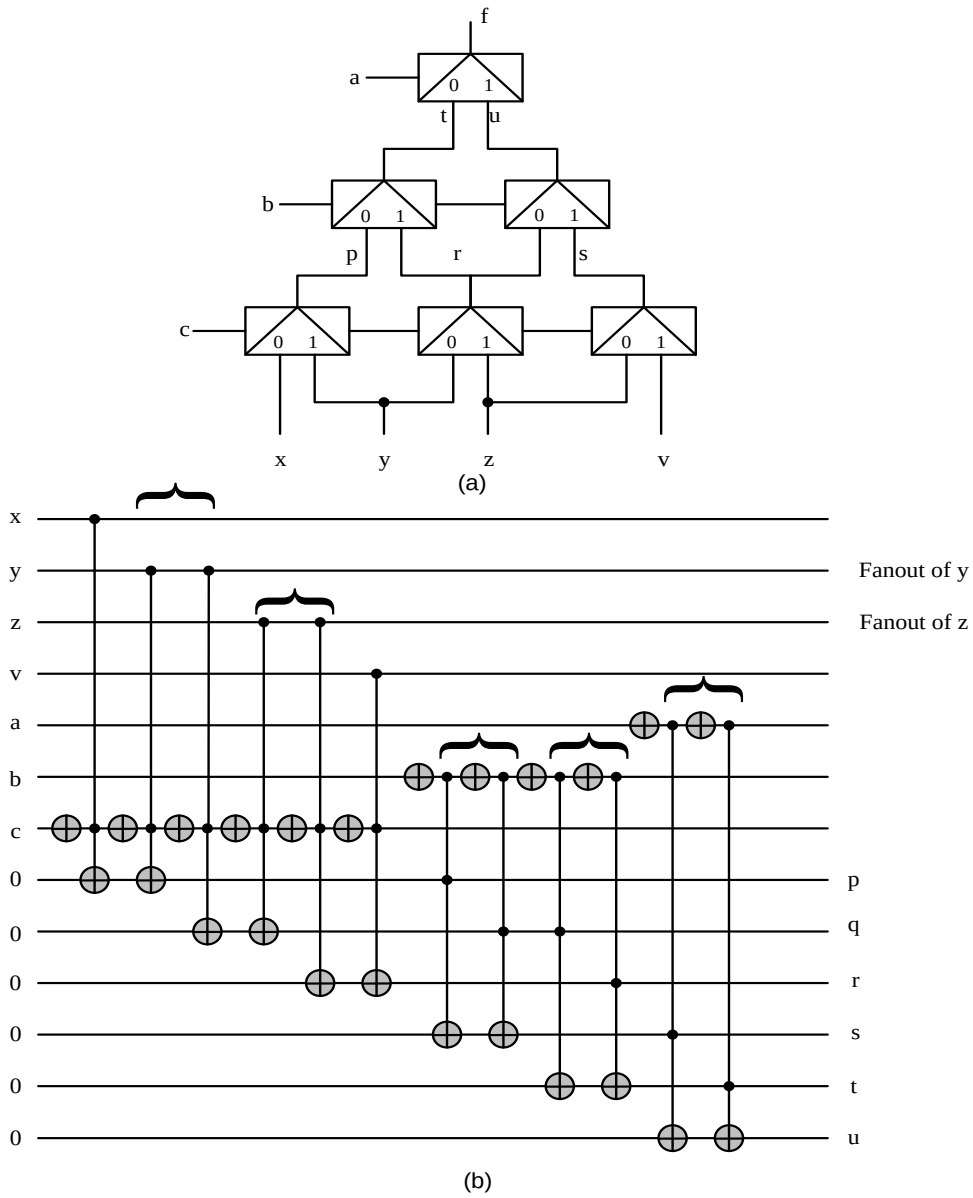


Figure 7.2. Lattice Diagrams in Quantum Array, (a) the standard Shannon Lattice, (b) its realization in quantum array without mirrors. Groups of gates corresponding to (non-scalable) multiplexers are shown. This figure demonstrates that fan-out can be realized without additional ancilla bits in some regular cases.

7.2. The concept and design of the Dipal gate.

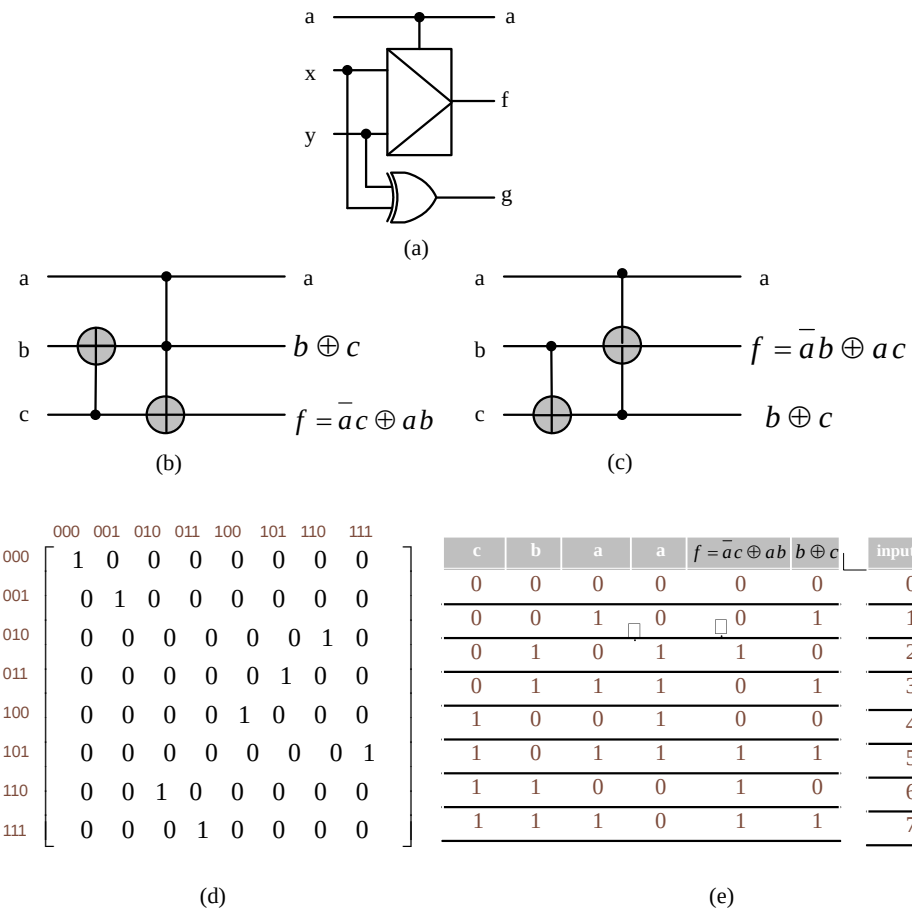
The Dipal Gate (now onwards referred as d-gate) (name given by my advisor Dr. Marek Perkowski) is a new invention and one of the important contributions of my research. This gate is never presented in the literature before. The basic d-gate is a reversible

counterpart of the classical multiplexer. The Boolean expression for the multiplexer can be presented as shown in Equation 7.1 This function can be represented in a positive polarity form by replacing $\bar{a} = 1 \oplus a$ and further simplified as shown in Equation 7.2. The reversible logic representation of Equation 7.2 is shown in Figure 7.3b.

$$f = \bar{a}b \oplus ac \qquad \text{(Equation 7.1)}$$

$$\begin{aligned} f &= [1 \oplus a]b \oplus ac \\ f &= b \oplus ab \oplus ac \\ f &= b \oplus a[b \oplus c] \end{aligned} \qquad \text{(Equation 7.2)}$$

Figure 7.3a shows the d-gate using a classical binary circuit. Actually the invention of the d-gate is very natural when this gate is drawn as a quantum array. As is well-known, the Fredkin gate in some quantum technologies such as Ion Trap and NMR is realized by surrounding the Toffoli gate with two Feynman gates. As regular diagrams can be created with Toffoli gate only and with Fredkin gate only, they can be created as well when we take a gate that is just half way between Toffoli gate and Fredkin gate – taking a Toffoli gate and just one Feynman gate preceding it. This leads to the invention of the d-gate, the schematic of the d-gate is shown in the Figure 7.3b. The other variation of d-gate for negated variable is presented in Figure 7.3c. For the completeness of the design of d-gate, the unitary matrix and the truth table for d-gate in Figure 7.3b is shown in Figure 7.3d and e respectively. Many variations can be created and can be used in creating a Lattice Diagram based quantum arrays as well as a layered diagrams. Let us now prove that this gate is a reversible gate. We calculate its inverse using Boolean Logic. The derivation is presented in Figure 7.4.



c	b	a	a	$f = \bar{a}c \oplus ab$	$b \oplus c$	input	output
0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1
0	1	0	1	1	0	2	6
0	1	1	1	0	1	3	5
1	0	0	1	0	0	4	4
1	0	1	1	1	1	5	7
1	1	0	0	1	0	6	2
1	1	1	0	1	1	7	3

Figure 7.3. (a) d-gate using traditional components, (b) representation of d-gate with Toffoli and Feynman gate, (c) d-gate with negated variable, (d) unitary matrix for d-gate from Figure 7.3b, (e)truth table for d-gate from Figure 7.3b.

$$g = x \oplus y$$
$$f = \bar{a}x \oplus ay$$
$$x = g \oplus y$$
$$ay = \bar{a}x \oplus f$$
$$ay = (1 \oplus a)(g \oplus y) \oplus f$$
$$ay = g \oplus y \oplus ay \oplus f$$
$$0 = g \oplus y \oplus ay \oplus f$$
$$y = g \oplus ay \oplus f = \bar{a}g \oplus f$$
$$x = g \oplus y = g \oplus \bar{a}y \oplus f = ag \oplus f$$

We obtain,

$$y = \bar{a}g \oplus f$$
$$x = ag \oplus f$$

Figure 7.4. Calculation of inverse gate to d-gate.

The powerful property of the d-gate is that many variants of the d-gate can be created by using negation on input/output nodes as well as using SWAP gates. Various d-gates are presented in Figure 7.5a. Note here that d-gates can be generalized as a one control bit gate as shown in Figure 7.5b. The control bit is repeated at the output and the transform operation T is performed on two target bits if control bit is set. In a similar manner a 3×3 Toffoli gate can be represented as variation of the d-gate, this is shown in Figure 7.5a.

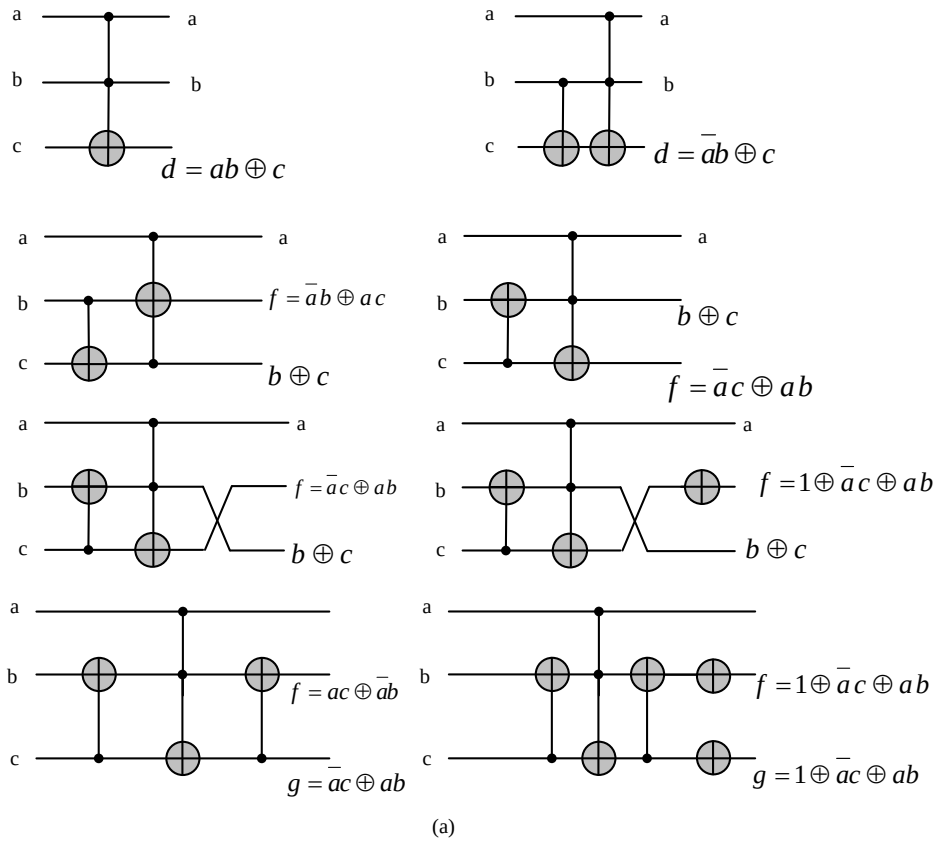


Figure 7.5 (a) Various d-gates (also known as d-gate family), (b) Representation generalized d-gate with one control line.

The expansion method that is applied for creating quantum array using d-gate is very similar to the one presented in the previous chapter but I do not know yet any good heuristic to select a particular d-gate type in every slot. The next section describes how to create the quantum array and the layered diagrams for Lattice Diagrams based synthesis methods.

7.3. Layered-based expansions with Toffoli gate and Dipal Gate Family.

The Positive Davio Lattice can be redrawn to a standard form of a quantum array as described in the Chapter 6. For instance, to help the reader, the Lattice Diagram from function F3(a, b, c) is presented in Figure 7.6a in a form that is intermediate between a Lattice Diagram and a quantum array. Then this diagram can be transformed as in Figure 7.6b, where every intersection of wires from Figure 7.6a is replaced by a SWAP gate as shown in Figure 7.6b. This way we get a new type of regular structure realized in quantum array with regular connections. The long connections that are typical for standard Toffoli gates are avoided. The whole trick was to use SWAP gates. Their number is however smaller because of regularity of the new structure from Figure 7.6a. The number of garbage qubits is the same as in standard quantum array and there are SWAP gates added, however there are no Toffoli gates realized on non-neighbor qubits. Although we add many SWAP gates, the cost of them is not restrictive as each such gate can be realized with 3 Feynman gates [Nielsen 00].

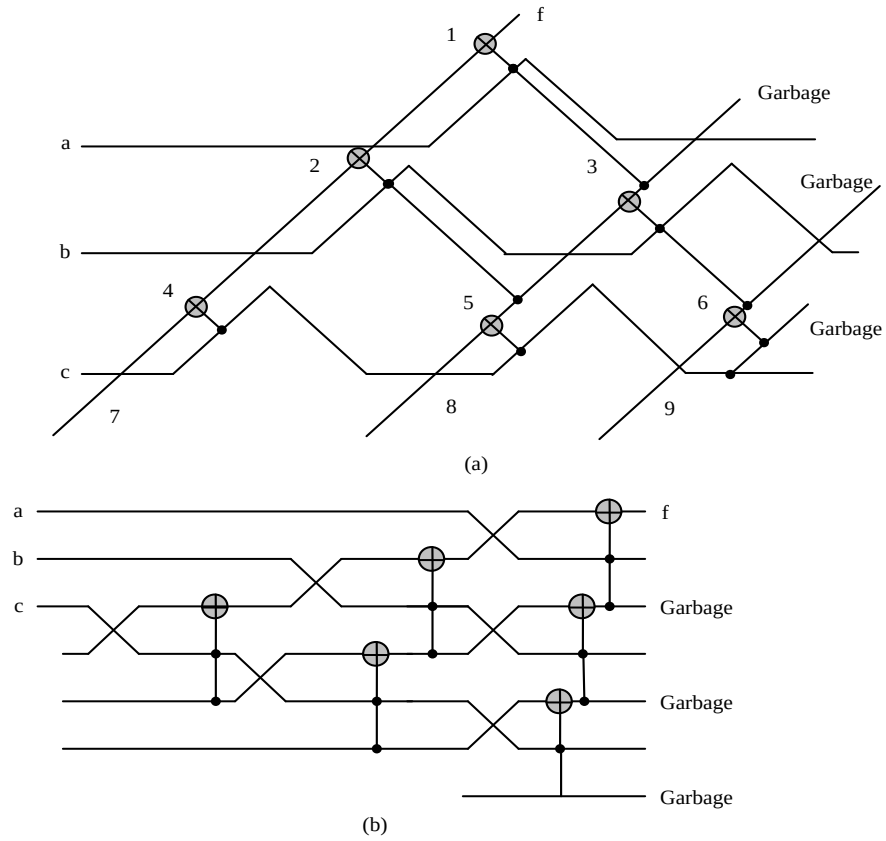


Figure 7.6. Transformation of function from classical Positive Davio Lattice to a quantum array with Toffoli and SWAP gates. Each SWAP gate is next replaced with 3 Feynman gate, (a) intermediate form, (b) final Quantum Array.

The Figure 7.7a presents the transformation of standard Shannon Lattice drawn in another way. A regular quantum array with addition of SWAP gates for Shannon Lattice is shown in Figure 7.7b. This figure is very useful as it explains that the Toffoli gate can be replaced by the d-gate and the method will apply to new types of diagrams, which I call the layered diagrams. This is illustrated in Figure 7.7c.

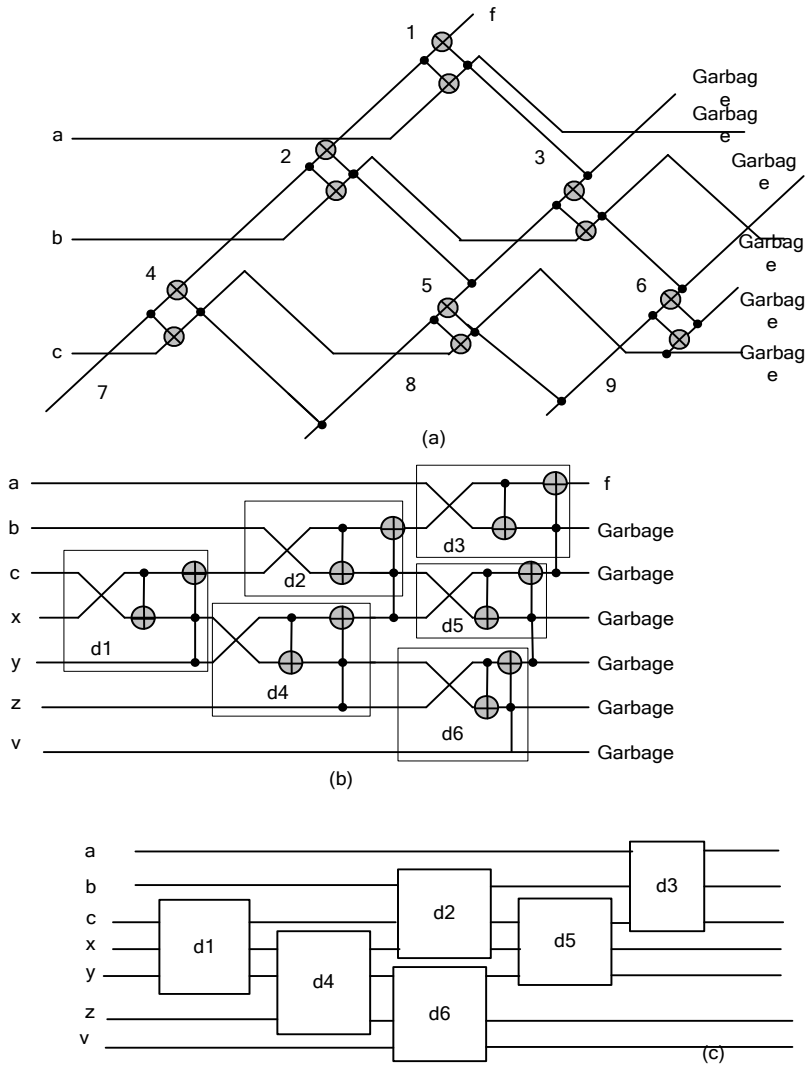


Figure 7.7. A view of regular layout with Dipal Gates. (a) the structure drawn similar to lattice diagrams, (b) the corresponding quantum array with SWAP gates added, (c) the general pattern abstracted from Figure 7.7b that uses d-gates and Swaps being parts of d-gate Family.

The Figures and transformations as shown above are useful as they explain that the Toffoli gate can be replaced by the d-gate and the method will apply to new types of diagrams, which is called the layered diagrams. This is illustrated in Figure 7.8. Also observe that in the same line of reasoning we can replace the d-gate by Fredkin gate and

use the same approach to synthesize reversible quantum circuits. This is illustrated in Figure 7.8.

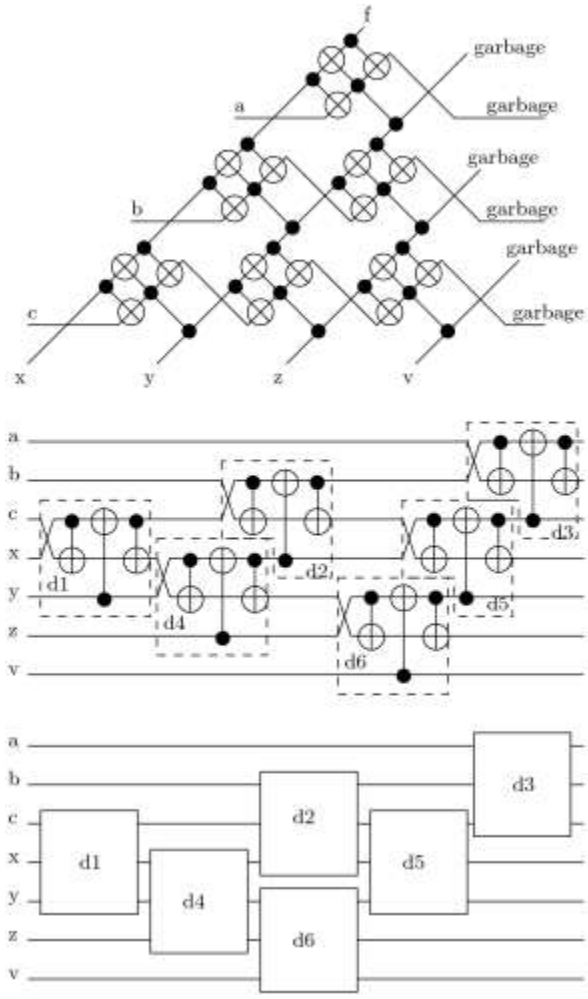


Figure 7.8. Another view of regular layout with Fredkin gates. (a) the structure drown similar to lattice diagrams, (b) the corresponding quantum array with SWAP gates added, (c) the general pattern abstracted from a lattice diagram similar to that from Fig. 7b that uses Fredkin Gates and SWAP gates being parts of a Fredkin Gates.

Figure 7.9 illustrates a general pattern and diagram of all d-gate Family expansions. Every rectangle represents variant of d-gate. Control variables are drawn as horizontal lines.

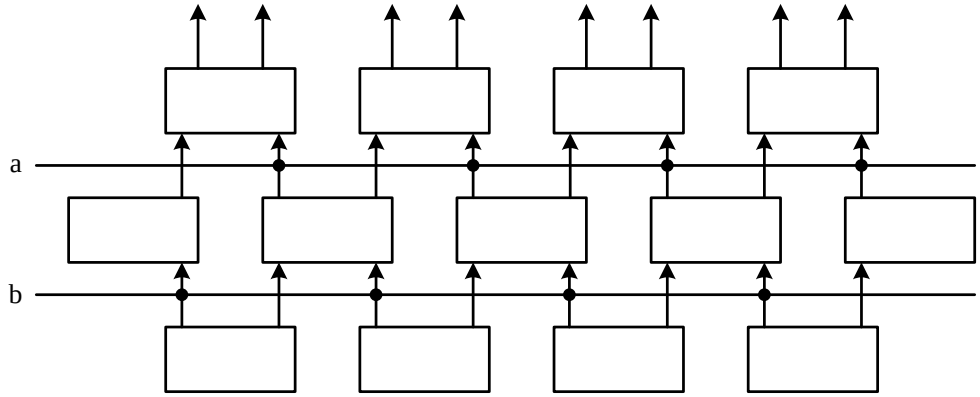


Figure 7.9. The general pattern of layered diagram using d-gate Family.

7.4. Algorithm pseudo code for creating the Shannon Lattice Diagrams.

The algorithm for crating Shannon Lattice Diagrams is described in this section. The algorithm pseudo code is similar to one shown in the Chapter 6 for the Davio Lattice and presented here for convenience. For future development I will advance my software to provide user a menu selection to choose amongst different Lattices. The input to the algorithm is a Boolean function and order of variables which will be used in expanding nodes at every level. The Boolean function is an output of a root node of the Lattice. The quantum array is created for the lattice diagram created using the algorithm as described in the section 6.5.

Input Boolean function;
Input Order of variables for a given Boolean function;

level = 1;
node_number = 1;
root_node = Input Boolean function;
variable = current_variable;
Add root_node as a first element of a linear linked list;

Create PCF (Positive cofactor) for a root_node using variable;
PCF.level = level;
PCF.node_number = 2×node_number;

```
        Create NCF (Negative cofactor) for a root_node using variable;
        NCF.level = level;
        NCF.node_number = (2×node_number) + 1;

        Simplyfy PCF and NCF to eliminate common terms;
        Add PCF and NCF of the root_node to the linear linked list (at the end of a
linked list);
        while (while all nodes in a Lattice are !constant)
        {
          variable = Next variable in the list;
          ++level;
          foreach (node in the level)
          {
            Create PCF using variable;
            PCF.level = level;
            PCF.node_number = 2×(node_number of a parent node);

            Create NCF using variable;
            NCF.level = level;
            NCF.node_number = (2×(node_number of a parent node)) + 1;

            Simplyfy PCF and NCF to eliminate common terms;
            Add PCF and NCF to the linear linked list;
            Traverse the linked list to go to the level in current iteration;
            Identify nodes eligible for performing joining operation in the current
level;
            Perform joining operations;
            Simplify the function of this new node to eliminate common terms;
            Delete nodes on which joining operation was performed;
            Add new node generated with appropriate node number;
          }//end Foreach
        }//end while, Lattice diagram is complete
//Display the Lattice diagram
        while (!end of the linked list)
        {
          Display Boolean expression in the node;
          Display variable used by the node;
          Display level;
          Display node_number;
      }//end while
```

Figure 7.9 Algorithm for implementation of the Shannon Lattice Diagrams.

7.5. Experimental Results.

The Table 7.1 shows experimental results for synthesized circuits using d-gate family. The number of gates and the quantum cost for the array created with d-gate family is shown in column five and six. The quantum costs of the reversible gates are computed as method used in contemporary algorithms [Maslov04]. The Table 7.1 also shows comparison of results for Shannon lattice based quantum circuits, Positive Davio based quantum circuits with DMM and AJ methods. The best quantum cost results are marked as italic and bold. Dashes in the Table mean no results are available or possible.

TABLE 7.1

Benchmark	#Inputs	#Gates pDv Lattice	Cost pDv Lattice	#Gates Shannon Lattice	Cost Shannon Lattice	#Gates DMM	Cost DMM	#Gates AJ	Cost AJ
2to5	5	31	107	41	117	15	107	20	***100***
rd32	3	4	***8***	4	8	4	***8***	4	***8***
rd53	5	11	***39***	18	46	16	75	13	116
3_17	3	10	21	15	26	6	***12***	6	14
6sym	10	19	75	27	84	20	***62***	NA	NA
5mod5	5	14	***58***	30	81	10	90	11	91
4mod5	4	6	18	12	24	5	***13***	5	***13***
Ham3	3	3	***7***	6	10	5	***7***	5	9
xor5	5	4	***4***	4	4	4	***4***	4	***4***
Xnor5	5	5	***5***	5	5	------	-------	-------	------
Decod24	4	10	***30***	20	40	------	-------	11	31
Cycle10_2	12	180	***860***	270	950	19	1198	-------	------
Ham7	7	22	***58***	32	68	23	81	24	6
Graycode6	6	5	***5***	5	5	5	***5***	5	***5***
Graycode10	10	9	***9***	9	9	9	***9***	9	***9***
Graycode20	20	19	***19***	19	19	19	***19***	19	***19***
nth_prime3_inc	3	4	***6***	6	8	4	***6***	-------	-----
nth_prime4_inc	4	16	***48***	29	61	12	58	-------	-----
nth_prime5_inc	5	29	91	39	101	26	***78***	-------	-----
Alu	5	5	17	10	22	------	-------	18	114
4_49	4	16	***52***	22	58	16	58	13	61
Hwb4	4	12	***28***	15	31	17	63	15	35
Hwb5	5	24	***96***	38	110	24	104	-------	-----
Hwb6	6	32	***128***	40	134	42	140	-------	-----
Pprm1	4	9	***33***	14	38	-----	------	-------	------

CHAPTER 8

From Binary to Multiple Valued Quantum Array.

8.1. Other Structures for Layered-based expansions.

The Figure 8.1 presents various regular structures that we considered for layered expansions. Observe that our general scheme allows creating more general expansions/inverse expansions and structures, all derived starting from the basic pattern of Figure 8.1a. By adding oblique local connection, the 2×2 array is converted to a 3×3 array from Figure 8.1b. As we will see, there are many 3×3 patterns, and for many of them we found interesting applications.

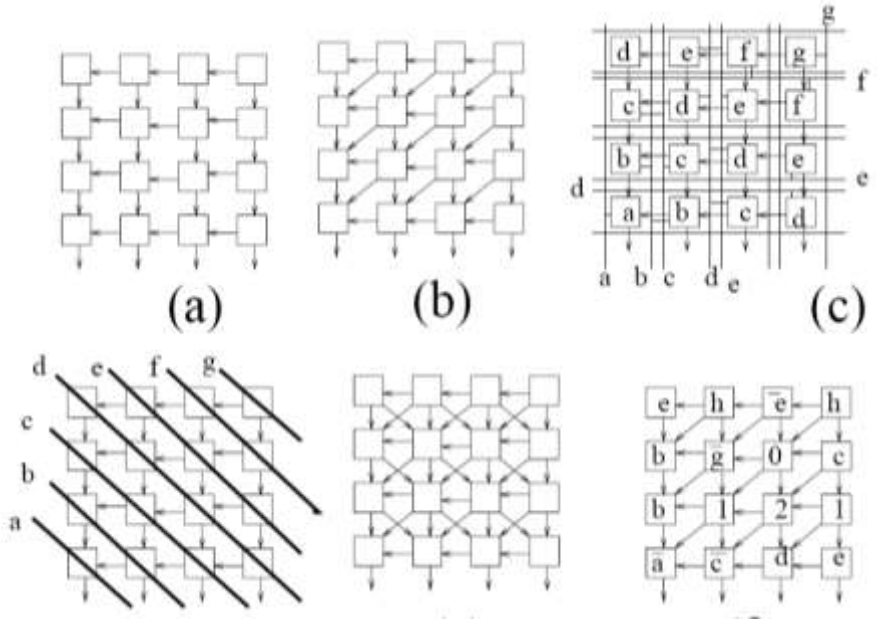


Figure 8.1. Examples of regular diagrams: (a) standard 2×2 array with only local connections, (b) standard 3×3 array with only local connections, (c) 2×2 array with two vertical and two horizontal buses per cell and the programming of its buses. Thus the grid is 6×6. (d) Akers Array with buses, 3×3 grid. These buses may be fat increasing the grid pattern size. (e) regular array with 4×4 grid and no buses. (f) a regular array with 3×3 grid and individual single-variable control of each cell. Irregular routing of these signals is not included to grid size calculation here.

Figure 8.1.c presents an example that for small arrays we do not need oblique buses, because cells can obtain control variables from vertical and horizontal buses. Here, two vertical and two horizontal buses are assumed, the same as the architecture from company Concurrent Logic. Figure 8.1d presents an example of our basic structure with oblique buses, which in this case is programmed to a (generalized) symmetric function, because there is no repetition of variables. This array with repeated variables; a, b, b, c, b, b, a would depict the Universal Akers Array. Figure 8.1e presents an example of basic structure with two oblique local connections, thus creating a 4×4 grid. Such cell can either execute Quaternary Shannon Expansions, or it executes standard Shannon expansion to two branches, and next selects any two of its successors to map the branches to them and execute corresponding inverse expansions. Finally, Figure 8.1f presents an example of a 3×3 structure in which each cell is programmed to a different control variable or a constant. Such array executes ternary expansions. For larger array it may be a problem with routing all control input signals.

Concluding this subsection, it should be obvious by analogy to standard diagrams, that with given order of (possibly repeated) variables these expansions are canonical if the same variable and expansion type is in every level. (Analogy to Functional Kronecker diagrams [Sarabi99]). By assuming special additional rules, also more general diagrams with mixed (Pseudo-Kronecker type) expansions could be made canonical, the same as the Pseudo-Kronecker decision diagrams, as presented by paper of Dr. Perkowski with his student Li Fei Wu [LiFei92].

8.2. Generalizations from binary to ternary.

In this chapter, we presented very general unified concepts of expansions, regular diagrams, and regular layouts for binary quantum circuits. We showed only some of their practical applications and restricted ourselves to binary circuits. We believe, however, that all these methods can be developed much more based on the broad fundamentals already given here and explained in a sufficient detail. The algorithms for creating Toffoli Lattices and Dipal Gate Family Layered Diagrams have been programmed by me and compared on binary benchmark functions.

The presented model of creating quantum combinational architectures is suitable for the realization of a wide class of quantum circuits; binary, multiple-valued and hybrid circuits. The ternary and quaternary will be presented in chapter 9. Before we do this, let us discuss how the results from this chapter were generalized. For each of the gates and expansions we found the generalization from binary to ternary. For instance, Figure 8.2 illustrates generalization of Toffoli and Feynman gates from binary to ternary. Although our thinking method is not seen because of various notations for binary and ternary circuits, the principles of these circuits are the same, so we can now extend these gates to any radix of logic.

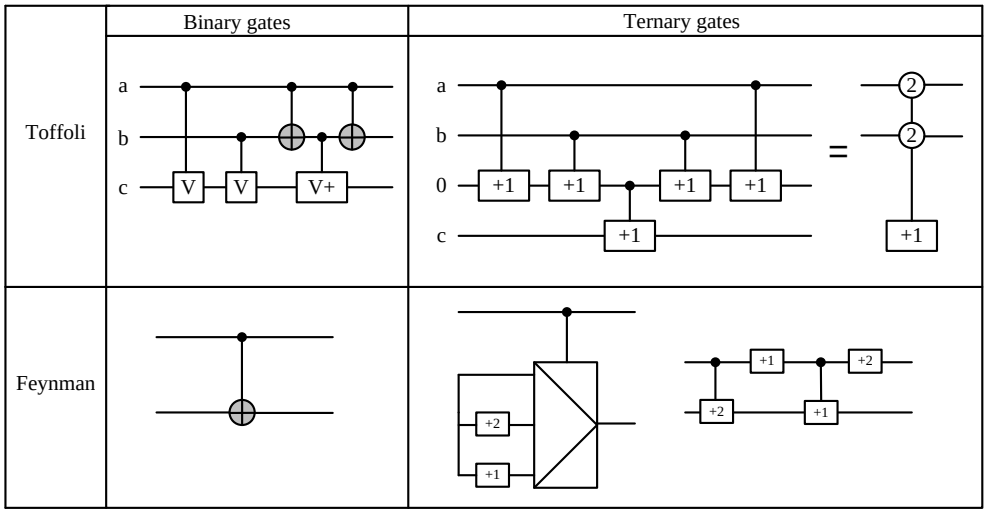


Figure 8.2. Comparison of binary and ternary gates used in expansions.

In chapters 6 and 7 I have introduced the hierarchical regular architecture model results from the general premise to use local signal interconnections whenever possible, and global signal interconnections only regularly (like in input busses) and only when absolutely necessary. It results also from attempts at **finding general tessellation structures** for two and three-dimensional space in which such structures can be practically realized in Ion Trap architectures. All these considerations are made recently practical for quantum circuits thanks to progress in 2D and 3D Ion Trap technology. Every gate should be mapped as a "supernode" (small regular graph) to a local neighborhood in a regular structure, in such a way that the locality of connection relation is preserved for all neighbor supernodes in the local neighborhood with the smallest number of ions in supernodes and their local connections (see the example from the end of chapter 4).

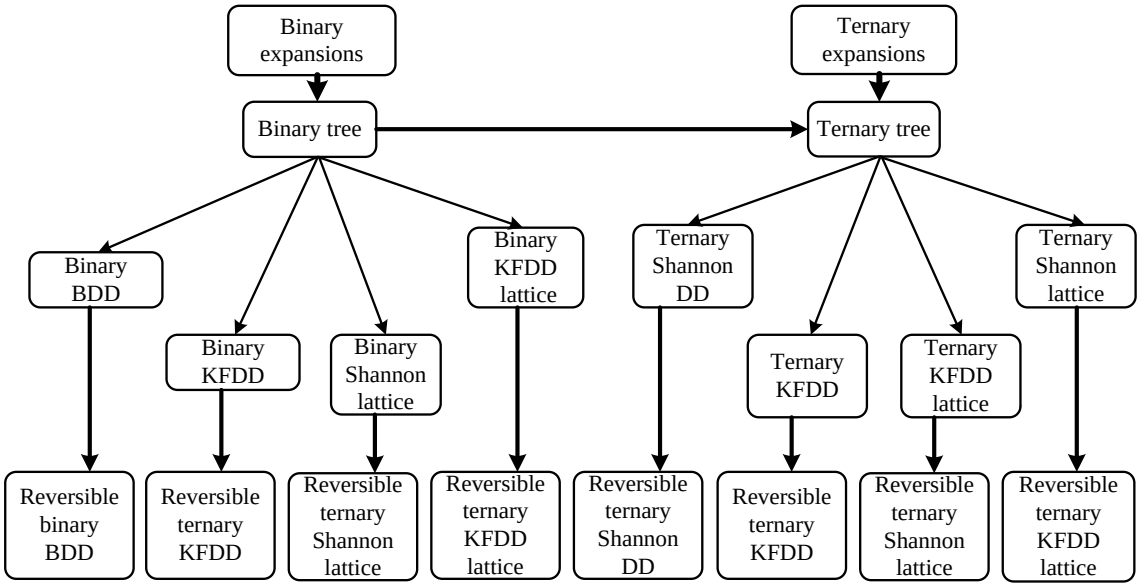


Figure 8.3. Generalizations of expansions, trees, and regular diagrams from binary to ternary.

Figure 8.3 illustrates the generalizations of the concepts from binary to ternary. We generalize first operators, next expansions that use these operators and finally the regular diagrams. Although the dissertation does note discuss every variant in detail, I know how each of them can be done. I have described however in chapters 5 and 6 sufficient detail that the careful reader should be convinced that each of these generalizations can be done in principle.

When we have all ternary generalizations of regular diagrams, they can be mapped to 1D, 2D or 3D layout spaces, based on various regular layouts (grid patterns) in each. These next mappings are shown schematically in Figure 8.4.

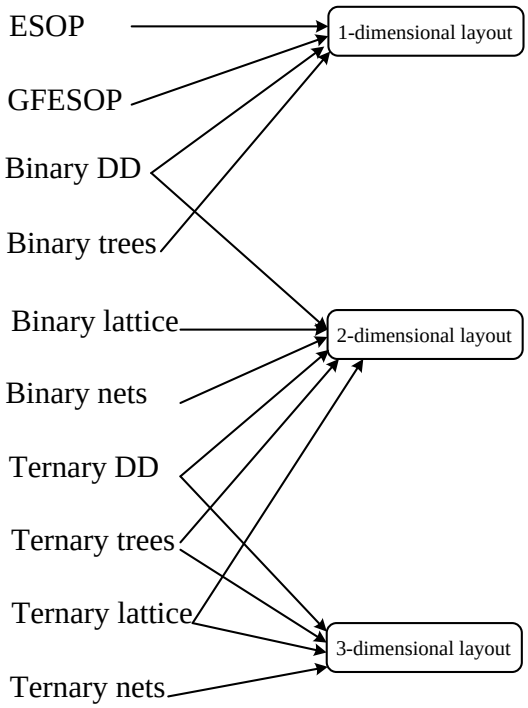


Figure 8.4. Mappings of various regular diagrams to one-, two- and three-dimensional regular layouts.

The main idea of the presented approach can be summarized as follows: starting from **all possible** neighbor geometries in two and three dimensional spaces, we create **all possible regular structures**. This is more powerful than in the previous structures [Akers72] which considered limited planar geometries. Also, we design the structure step-by-step in the process of expansions and reverse expansions until functions are trivial. In contrast, Akers creates the worst-case structure from the scratch, which is **extremely** wasteful for most functions. On nearly all of investigated by us functions our areas and numbers of cells were much better than those of Akers. Also, Akers did not show methods for Davio, multi-valued and, which makes our approach much more general. Similarly, we can show that our approach is more general than PLAs or other known structures. Next we design **arbitrary expansions** for any of the structures. New expansions can be

constructed based on the orthogonal logic approaches, or any other canonical or non-canonical function expansions. We demonstrated also the very high number of **various new expansions**, in contrast to only Shannon expansion types used by Akers [Akers72] and only Positive Davio, Negative Davio, Shannon and Rotated Shannon expansions used in works of Jeske, Perkowski, Drechsler, Falkowski, Marek-Sadowska and other researchers who worked in the past on lattice diagrams and similar concepts.

Concluding, the previous chapters and this chapter demonstrated that the realizations based on orthogonal expansions as well as more general ones, based on sets of **not necessarily orthogonal functions**, lead to regular diagrams which can be easily mapped to various types of quantum regular structures in Ion Trap. The proposed particular binary variants are excellent tools for fast prototyping of binary quantum circuits in various regular structures. This dissertation provides the researchers in the field with an opportunity to experiment with hardware realizations of various logic circuits using quantum simulators and my quantum synthesis CAD tools that I have developed. The presented examples demonstrate simplicity of realization of a wide class of such circuits, which will also enable the implementation of design automation procedures and concrete CAD tools for quantum circuits based on regular data flows and expansions.

CHAPTER 9

Galois Field Logic, Group Logic and Their use for Regular Quantum Circuit Synthesis.

In chapter 3 I introduced basic trees, diagrams and forms and in chapter 5 I introduced new lattice diagrams and other diagrams for binary quantum logics. However, the set of concepts useful in 2D and 3D Ion Trap design is not exhausted by these two approaches. In this chapter we will try to generalize all my ideas from chapters 5-8 from binary to multiple-valued logics. As we will see, it can be done in many ways. While the conceptual work in chapters 5-8 for binary logic has been completed, the Chapter 6 is not completed in the sense that there are many expansions that we did not analyze yet in full detail. Chapter 5-8 gives us however the blueprint and direction for future work. Below, in section 9.1 I present one more classical form that is the base for the new developments in this dissertation. More new gates, expansions, and regular diagrams with their mappings to regular structures will be discussed later in the chapter.

Figure 9.1 presents the hierarchical method to design new regular layouts. We create first various synthesis methods to synthesize simple gates such as operators and expansions. Some methods are taken from the literature. Thus I create blocks for higher level design, so that this higher level design does not have to concern itself with the low–level gate design details. A useful abstraction for both quantum logic synthesis and quantum block-level design is a Quantum Multiplexer introduced by Dr. Perkowski [Khan05a, Khan05b, Khan05c]. Based on the concept of Ternary Quantum Multiplexer I was able to design various operators, expansions and gates using Muthukrishnan-Stroud gates as their

physical realization. It is interesting to note that although all our gates for both the Galois Fields and other algebras are built on top of Muthukrishnan-Stroud (MS) gates, we never use the MS gates directly for synthesis. I tried to do this but was unable to find a method. Ternary MS gates are built internally with single-qubit rotation gates and interaction gates. The hierarchy from Figure 9.1 can be analyzed from top and from bottom. Analysis from top, tells us how the concepts were created historically. Using it from bottom can allow us to build in future new gates and methods starting from the lowest-level primitives, that are directly realizable in hardware. This is not a topic of this dissertation, however.

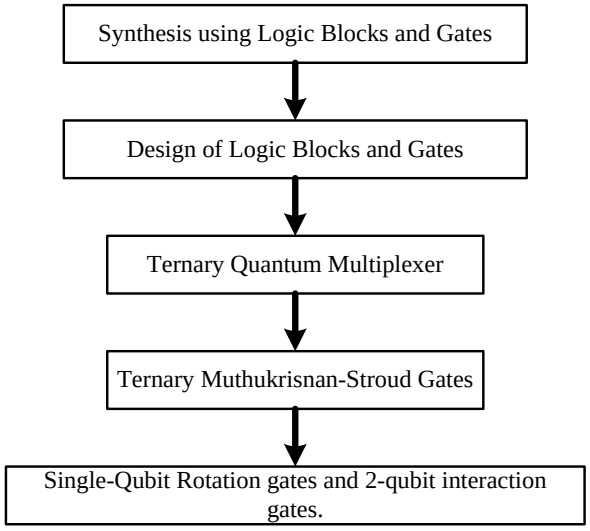


Figure 9.1. Hierarchical Decomposition and Synthesis of Ternary gates. This figure shows the plan of our research in this chapter.

9.1. Binary Generalized Reed-Muller Forms and Classical Binary AND-EXOR Hierarchy.

As may be recalled from literature [Debnath95, Debnath96, Dill97], while more restrictive than the Exclusive-Or Sum-of-Products (ESOP) expression, the GRM equation

form incorporates the Fixed-Polarity Reed-Muller (FPRM) and Positive Polarity Reed-Muller (PPRM) Forms as its special cases.

Definition 9.1.1:

The Generalized Reed-Muller Form (GRM) is a general, canonical expression of the Exclusive-Or-Sum-of-Products type, in which for every subset of input variables, there exists at most one term with any arbitrary polarities of all variables. Thus for an n-variable function there are $n2^{n-1}$ literals and $2n^{2^{(n-1)}}$ polarities. The GRM expansion of an n-variable function is shown as in Equation 9.1.1.

$$f(x_1, x_2, .. x_n) = a_0 \oplus a_1 x_1 \oplus .. \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus .. \oplus a_{n(n-1)} x_n x_{n-1} \oplus .. \oplus a_{12...n} x_1 x_2 ... x_n$$

(Equation 9.1.1)

where,

X can be expressed either as a positive literal x_i or a negative literal x_i'

X = (X □ □

□ = 0/1 for positive/negative polarity

a_i = coefficient of X_i, and can be 0 or 1

Example 9.1.1:

An example of a GRM is given as given in Equation 9.1.1:

$$f(x_1, x_2, x_3) = x_1' \oplus x_1 x_2 \oplus x_3' \oplus x_1 x_2 x_3 \qquad \text{(Equation 9.1.2)}$$

Observe that variable x_1 is negated in term x_1' and positive in term $x_1x_2x_3$. The corresponding circuit is shown in classical notation in Figure 9.1.1a and in quantum array notation in Figure 9.1.1b.

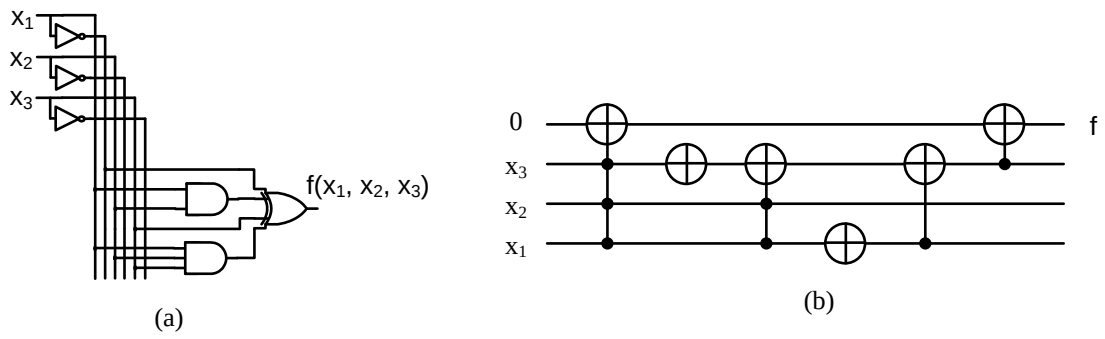


Figure 9.1.1. (a) Classical logic diagram of GRM Form for function from Example 9.1.1, (b) Quantum array for the same form.

Finally, the most general classification of AND-EXOR equations, including the PPRM, FPRM, and GRM classification forms, is the Exclusive-Or-Sum-of-Products (ESOP) expression (not a form, not canonical!). While the ESOP form is not a canonical expression, there are no restrictions on the terms. Defined loosely, it is an expression simply consisting of arbitrary terms combined with the EXOR operation.

Definition 9.1.2: The Exclusive-Or-Sum-of-Products expression (ESOP) is a non-canonical form in which arbitrary product terms are combined with EXOR logic gates.

The relations between the classical, binary Reed-Muller expressions are shown in Figure 9.1.2. The forms illustrated in this diagram have the following relations:

(1) PPRM $\subset$ FPRM,

(2) FPRM $\subset$ PSDRM,

(3) FPRM $\subset$ KRO,

(4) KRO $\subset$ PSDKRO,

(5) PSDRM $\subset$ PSDKRO,

(6) PSDKRO $\subset$ FKRM,

(7) PSDRM $\subset$ GRM.

(Note that the GRM Form, while still canonical, usually requires nearly as few terms as the ESOP Form.) The forms shown in dashed lines, FKRM and ESOP, are not canonical.

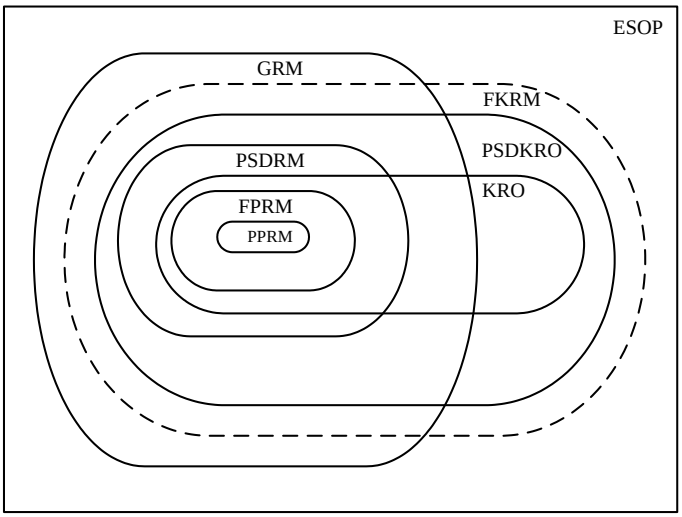


Figure 9.1.2: All AND-EXOR Forms (non-canonical forms indicated with a dashed line).

A subset of the Reed-Muller Hierarchy is given with corresponding expansions, trees, decision diagrams, and forms in Table 9.1.1.

The Galois Field Logical Hierarchy is a logical family in which trees, decision diagrams and forms are based on the expansion of an AND-EXOR canonical form with Shannon, Positive and Negative Davio Expansions and in which the mathematical properties of a Galois Field hold.

9.2. Binary ESOP Logic.

While not as widely utilized for integrated circuit design as the AND-OR Sum-of-Product (SOP) logic, the exclusive-or sum-of-product (ESOP) form offers high flexibility paired

together with the benefits offered by AND-EXOR logic. This analysis was made, encouraging future design development with ESOP logic, as follows [Song93]. The functions realized by such circuits can have fewer gates, fewer connections, and take up less area in VLSI and especially, FPGA realizations. They are also easily testable [Fujiwara86, Pradhan87]. It was shown, both theoretically and experimentally [Perkowski99a, Perkowski99b] that ESOPs have on an average smaller numbers of terms for both "worst case" and "average" Boolean functions. It was also shown that ESOPs and all their sub-families have their counterparts in logic with multiple-valued inputs: Multiple-valued Input ESOPs (MIESOPS) [Sasao90a, Sasao91], Multiple-valued Input Generalized Reed-Muller forms [Schaefer91], Multiple-valued Input Kronecker Reed-Muller forms (MIKRMs) [Schaefer93], Multiple-valued Input Generalized Reed-Muller Trees (MIGRMTs) and others [Song93, Stankovic97]. Logic with multiple-valued inputs (mv logic, for short) generalizes the classical Boolean logic and finds many important applications in logic design. MIESOPs are never worse than ESOPs, and they were shown to be superior on several classes of functions. Here we want to investigate relations of these concepts to quantum circuits, and especially regular quantum circuits.

Table 9.1.1: Classical, Reed-Muller binary trees, diagrams and expansions.

Expansion	Tree	Diagram	Form
Shannon Expansion (S)	Shannon Tree	Binary Decision Diagram (BDD)	Sum-of-Product Canonical Form
Positive Davio Expansion (pD)	Positive Davio Tree	Functional Decision Diagram (FDD)	Positive Polarity Reed-Muller Form (PPRM)
Shannon, Positive and Negative Davio (S, pD, nD) (But only one type of expan-sion per level)	Kronecker Tree	Kronecker Decision Diagram (KDD)	Kronecker Reed-Muller Form (KRM)
Shannon, Positive, and Negative Davio (S, pD, nD) (But any subset in every level)	Pseudo-Kronecker Tree	Pseudo-Kronecker Decision Diagram (PKDD)	Pseudo-Kronecker Reed-Muller Form (PKRM)
Shannon, Positive, and Negative Davio (S, pD, nD) (No order of Variables)	Free Kronecker Tree	Free Kronecker Decision Diagram (FKDD)	Free Kronecker Reed-Muller Form

Previously, one of the major drawbacks to utilizing AND-EXOR logic was that function minimization was very difficult. Exact algorithms are intensively time consuming, while heuristic approaches have been limited in both application and quality. With the development of EXORCISM-MV-2, a software package providing "efficient minimization of arbitrary ESOP expressions for multiple-output, multiple-valued input, incompletely specified functions" [Perkowski89], future mapping to Ion Trap quantum

technology may become more practical. In addition to having a very general form, the ESOP has a two-level circuit implementation, which is easily testable. A function expressed in an ESOP equation usually requires fewer gates than that of other AND-EXOR forms and can never require more. Further, it has been shown that the ESOP theory can be extended to a multiple-valued logic; for instance, the Galois Field Logic.

9.3. Galois Field Logic.

9.3.1. Definitions, Literals and basic concepts.

As previously discussed, to incorporate the multi-valued logic concepts in quantum circuits, the Reed-Muller form may also be extended for Galois Fields [Dill97a, Dill97b]. The Galois theory demonstrates that for every number k^n, where k is prime and $n \geq 2$, there exists a unique matrix under mathematical operations, such that all group theory properties are satisfied. Hence, the same technique can be utilized for the representation of both binary and multi-valued logic. The relationship between the Reed-Muller Hierarchy and Galois Fields provides the capability of expressing multi-valued logic with decision diagrams and compact algebraic structures (or forms), which can be represented with "universal" expansion circuits. Such circuits can then be mapped to structural binary quantum technology [Dill98], as will be shown below.

In the Galois Field GF(n) of prime numbers the operations are modulo n addition and multiplication. Galois Field GF(2) and GF(3) are obtained with mod 2 and mod 3 operations, respectively. Figure 9.3.1 shows these addition and multiplication tables for GF(2). Figure 9.3.2. shows addition (addition modulo 3) and multiplication (multiplication modulo 3) operations for GF(3).

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

Figure 9.3.1. Addition (EXOR) and multiplication (AND) tables for GF(2).

⊕	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

×	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Figure 9.3.2. Addition and multiplication tables for GF(3).

Notice (in Figures 9.3.1 and 9.3.2) that each row and column contains every element, with the exception of the multiplication by zero operation. This property in the Galois mathematical operations tables is called a Latin Square and it exists for all Galois Field, GF(n), operations. The addition operations in GF(2) and GF(3) are examples of group operations. The reader can check all axioms of a mathematical group in these tables.

In the Galois Field GF(k) of non-prime numbers, the operation of *mod k* will not satisfy the Galois Field Theory axioms. In this case, the concept of field extension can be used to obtain a Galois Field of non-prime numbers. This may be necessary when it is desired that a set of operations demonstrates all of the Galois group and field properties. The following definition details the extension field concept [Dill97b].

Definition 9.3.1: A Field K can be extended to include elements $r_1, r_2, \ldots, r_n$, such that an extended field F exists, where field K and the new elements $r_1, r_2, \ldots, r_n$ together form a new field, $F = K(r_1, r_2, \ldots, r_n)$ where field K and $r_1, r_2, \ldots, r_n \in F$. Thus, the resultant field F is called the extension field of K if it satisfies all the Galois group and field properties.

Examples of the technique of field extension over finite fields are now given.

Example 9.3.1: The Galois Field GF(4) can be obtained by the field extension of the prime field GF(2), with an element (δ=2) such that the resulting field satisfies all the group and field properties. Figure 9.3.3 shows a GF(4) formed by the field extension of GF(2), with element "2", such that the group consists of the elements (0, 1, δ, δ+1) = (0, 1, 2, 2+1), which is equivalent to (0, 1, 2, 3). Notice that in the addition and multiplication tables, that the identity elements are "0" and "1", respectively. Further, observe that in these operations tables, each row and column contains every element in the group, thus demonstrating a Latin Square (with the exception of the multiplication by zero operation). Therefore as the original field is extended, the new field maintains the Galois Field properties.

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

(a)

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

(b)

Figure 9.3.3 (a) Table for Addition (EXOR-like group operations) in GF(4), (b) Table for Multiplication in GF(4).

Various decision diagrams used for switching functions can be uniformly regarded as graphical representations related to AND-EXOR expressions, derived by considering the switching functions as functions in the Galois Field, GF(2) [Kalay99c]. This is a very powerful link between Boolean algebra and GF algebra that my dissertation investigates. This can be better understood with the following examples:

Example 9.3.3: A GF-PPRM, in GF(2), is generated by the application of the Positive Davio Expansion, (i.e. all literals, x_i's have positive polarity). For binary logic using three variables, an example is given.

$$f(x_1, x_2, x_3) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_4x_1x_2 \oplus a_5x_1x_3 \oplus a_6x_2x_3 \oplus a_7x_1x_2x_3$$

Example 9.3.4: A GF-GRM, in GF(2), has both positive and negative polarities. For binary logic using three variables, an example is given.

$$f(x_1, x_2, x_3) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_4x_1x_2 \oplus a_5x_1x_3 \oplus a_6x_2x_3 \oplus a_7x_1x_2x_3$$

Where, x = x or x'

In addition to being standard Reed-Muller forms, the expressions in Examples 9.3.4 and 9.3.5 are actually polynomial forms in GF(2) for three variables. In this light, they illustrate the extension of the classical Reed-Muller logic forms to Galois Field forms. Further, the Galois Field concepts may be applied to irreducible polynomial expressions, over a field F, with the algebraic form $f(x) = a_nx^n + a_{n-1}x^{n-1} + \ldots + a_2x^2 + a_1x^1 + a_0$ on an indeterminate x, where $a_n, a_{n-1}, \ldots, a_2, a_1, a_0 \in F$ and n denotes the size of the Galois Field. Then, for GF(2) the single-variable polynomial has the form $f(x) = a + bx$, in GF(3) the single-variable polynomial has the form $f(x) = a + bx + cx^2$, etc. With these ideas of combining Galois Fields and irreducible, algebraic, polynomial forms, the Galois Field counterpart to Reed-Muller Expansions can be derived, which utilizes multi-valued logic. This concept is essential to the full development of the Galois Field Logic Hierarchy and next to regular structures introduced in this dissertation.

Thus, it is evident that in contrast to Boolean Logic, for a multi-valued logic in Galois Fields, there are a number of basic functions of a variable, such as exponents and inversions (or shifts). Further, these operations may be combined. Post Literals may also be applied. (Recall that a Post Literal is represented with a single superscript value to its left, denoting the input for which the maximum MVL element value is produced. For example, for ternary logic, with elements (0, 1, 2), the Post literal ${}^{0}x$ takes the value of 2 for x=0 and otherwise has the value of 0.) These applications of the possible operations for a single variable in GF(3) are graphically illustrated in Figure 9.3.8.

As previously alluded, in GF(3) the single-variable basic polynomials are x and x^2, where x can have two inverses and three possible logic states (none-basic polynomials are $2x + x^2$, $x + 2x^2 + 1$, etc). The elements 0, 1, and 2 represent the ternary logic states in GF(3). Thus, the variable x can have two possible operations: x' and x'', where the prime (inversion) superscript serves to describe literal functions on argument x, not to specify the logic states, i.e. any x can have any ternary value. These inversion operations are also referred to as "shifts". For a Galois Field of size n, GF(n), there are n-1 inverters. For example in GF(3), there are two inverters. The table in Figure 9.3.6 shows the shift operations between the variables.

Input	Inversion 1	Inversion 2
x	x'	x''
0	1	2
1	2	0
2	0	1

Figure 9.3.6 Inversions in GF(3).

The values within any Galois Field are related to each other and their various inverse operations with the addition or EXOR operation (generalized for multi-valued logic). For example in GF(3), the variable x can have three possible logic states and all logic states can be obtained by the "⊕ 1" operation. This is shown in Figure 9.3.7.

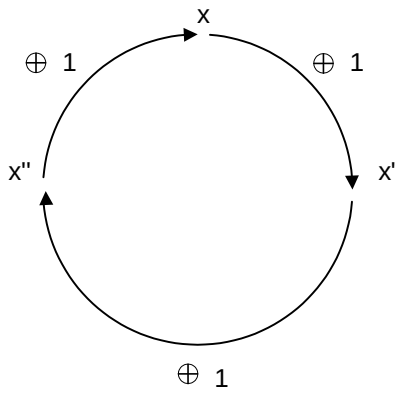


Figure 9.3.7. Explanation of the circular property of shift gates in ternary logic.

(Conventionally these operations are performed in a clockwise direction, however, both directions are logically correct.) The basic relations between the variables, shown in Figure 9.3.9, that describes the shift operations, are here given algebraically.

$$x \oplus 1 = x'$$

$$x' \oplus 1 = x''$$

$$x'' \oplus 1 = x$$

A graphical representation of the one and two position shifts is shown in Figures 9.3.6 and 9.3.7.

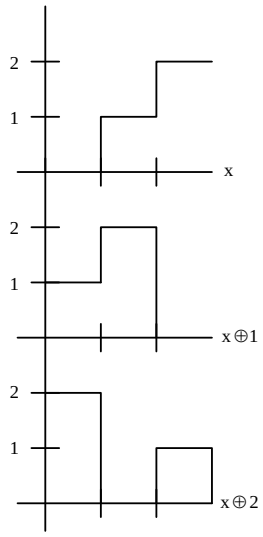


Figure 9.3.8 GF(3) variable x shown with shift (inversion) operation.

The GF(3) mathematical relations shown graphically in Figure 9.3.8 above, are given as follows:

For x = 0: x ⊕ 1 = 1 x ⊕ 2 = 2

x = 1: x ⊕ 1 = 2 x ⊕ 2 = 0

x = 2: x ⊕ 1 = 0 x ⊕ 2 = 1

An exponential operation, producing an x^2 term is also necessary for GF(3). These relations utilize the standard GF(3) multiplication tables given in Figure 9.3.4 and are graphically shown in Figure 9.3.9.

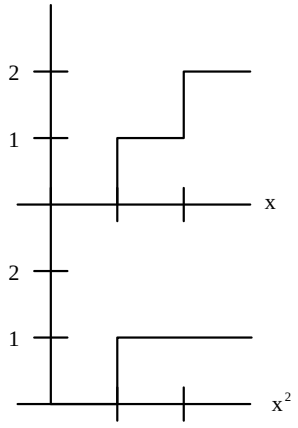


Figure 9.3.9: GF(3) variable x shown with and without exponent.

For convenience, the GF(3) mathematical relations shown graphically in Figure 9.3.9 above, are listed as follows:

For x = 0: $x^2 = 0 \times 0 = 0$

x = 1: $x^2 = 1 \times 1 = 1$

x = 2: $x^2 = 2 \times 2 = 1$

Post Literals may also be utilized in a Galois Field implementation. For GF(3), Post Literals of height one and two are shown in Figure 9.3.10 below. Utilizing the properties previously described, a Galois Field of size n, GF(n), can be derived for any multi-valued logic. Further, this logic is complete and realizable with only the mathematical Galois Field operations of addition, multiplication, and $n-1$ unique inverters (shifts). As we remember from Chapter 2 there are six reversible functions of single variable. As one of them, the identity, is not useful, five useful operations remain. Out of these five functions two; $x \oplus_3 1 = x'$ and $x \oplus_3 2 = x''$ are explained above. With this background, Galois Fields can now be fully related to Reed-Muller Logic.

9.3.2. Detailed analysis of binary expansions as our research base.

Most central to the development of Reed-Muller logic forms, the classical Shannon Expansion utilizes a variable polarity separation technique to represent a function. The Shannon Expansion for a variable x is obtained by splitting the variable into two different polarities, x=1 and x=0. The relation between these polarities can be represented as x = 1 $\oplus$ x'. For a binary function $f(x_1, x_2, \ldots, x_n)$ the Shannon Expansion, originally developed by Boole was already presented in Equation 9.3.2. It is reviewed here again:

$f(x_1, \ldots, x_n) = x_1' f(x_1 = 0, x_2, x_3, \ldots, x_n) \oplus x_1 f(x_1 = 1, x_2, x_3, \ldots, x_n)$ (Equation 9.3.1)

$f(x_1, \ldots, x_n) = x'f_0 \oplus xf_1$ (Equation 9.3.2)

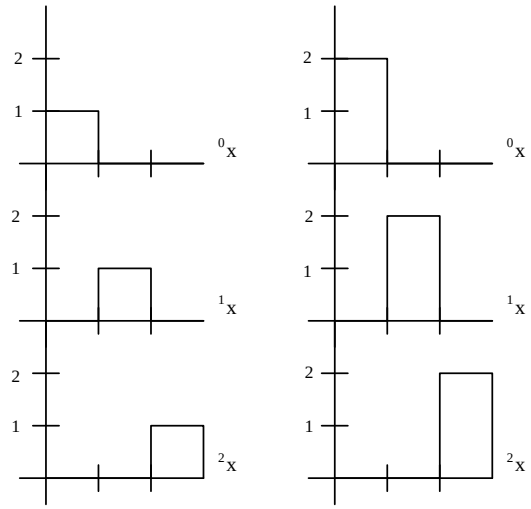


Figure 9.3.10: Post Literals of height 1 (left) and 2 (right). The first will be also called the reduced Post Literal.

Relating the Shannon Expansion to a K-map, another perspective can be gained about its application. This gives a visual depiction of how the components "fit" together to make the total function. In Figure 9.3.11, a simple K-map is given, with binary values represented by variables, with subscripts labeled for their location.

x \ y	0	1
0	f_{00}	f_{01}
1	f_{10}	f_{11}

Figure 9.3.11.

In Equation 9.3.1 for the Shannon Expansion, f_0 and f_1 are simply rows of the K-map, where x = 0 and x = 1, respectively. These are given in Figure 9.3.12.

$f_0 = f_{x=0} =$ | f_{00} | f_{01} |

$f_1 = f_{x=1} =$ | f_{10} | f_{11} |

Figure 9.3.12.

Starting with the Shannon Expansion, then, the K-map is related as follows in Figure 9.3.13.

$f(x) = x'f_0 \oplus xf_1$ {Shannon Expansion, GF(2)}

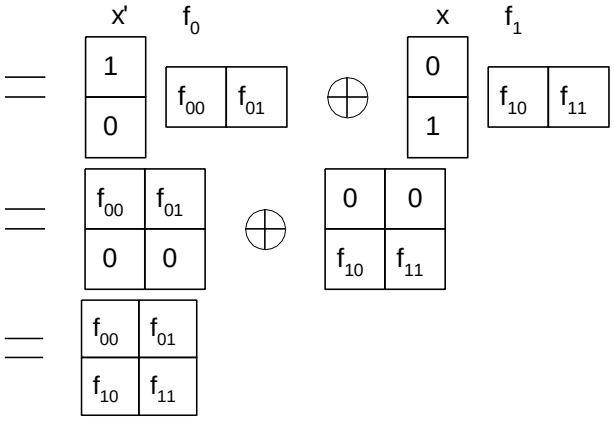


Figure 9.3.13.

The Shannon Expansion shown in algebraic form can also be represented as a decision tree. This is shown in Figure 9.3.14. (Here Post Literals, denoted with a left superscript, of height 1 are shown, which in <u>only</u> the binary case, are equivalent to the Boolean variables.)

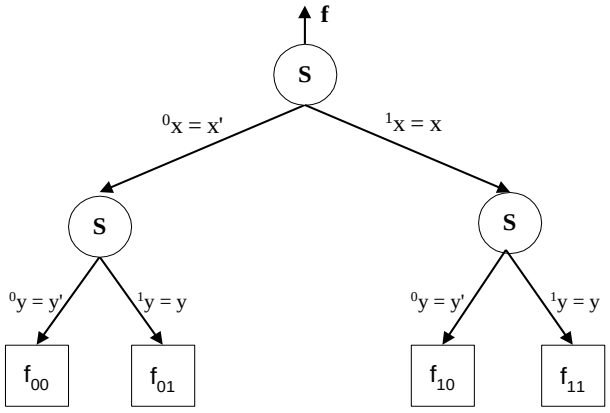


Figure 9.3.14 Shannon Tree for binary logic of two variables.

The Davio Expansions in binary logic are well known and derived from the Shannon Expansion by considering either the positive polarity (x) or negative polarity (x') of the variable x. Alternatively, starting from either the Positive Davio or Negative Davio, the Shannon Expansion may be derived. These derivations are shown in Equations 9.3.3 – 9.3.4 below. The Shannon, Positive Davio, and Negative Davio Expansions may be utilized to derive all possible expansions, to obtain all logic family forms, trees, and decision diagrams.

Derivation of Positive Davio:

Shannon $f(x)= x'f_0 \oplus xf_1$

By substituting $x' = x \oplus 1$

$$f(x)= (x \oplus 1)f_0 \oplus xf_1$$

$$f(x)= xf_0 \oplus f_0 \oplus xf_1$$

$$f(x)= x(f_0 \oplus f_1) \oplus f_0$$

Positive Davio: $f(x)= x(f_0 \oplus f_1) \oplus f_0$

Equation 9.3.4: The Positive Davio Expansion for binary (GF(2)) logic

$f(x) = x(f_0 \oplus f_1) \oplus f_0$

Derivation of Negative Davio:

Shannon $f(x)= x'f_0 \oplus xf_1$

By substituting $x = x' \oplus 1$

$$f(x)= x'f_0 \oplus (x' \oplus 1)f_1$$

$$f(x) = x'f_0 \oplus x'f_1 \oplus f_1$$

$$f(x) = x'(f_0 \oplus f_1) \oplus f_1$$

Negative Davio: $f(x)= x'(f_0 \oplus f_1) \oplus f_1$

Equation 9.3.5:

The Negative Davio Expansion for binary (GF(2)) logic

$$f(x) = x'(f_0 \oplus f_1) \oplus f_1$$

As we see, we use elementary Boolean algebra axioms and manipulation methods to derive expansions and circuits. The same is true in the multiple valued case but the technical transformations are more involved. Because I want the dissertation to be formal, all the steps of derivations are shown in many cases. In other cases they are not given as they can be done in a very similar way to the cases presented in full detail.

9.3.3. Derivation of Davio Expansions for the ternary case.

Using the method presented for the binary case, along these lines of thought, for a ternary logic in GF(3), the Shannon Expansion would naturally (by guessing) be as follows:

GF(2): $f(x) = x'f_0 \oplus xf_1$ {binary logic}

GF(3): $f(x) = x'f_0 \oplus xf_1 \oplus x''f_2$ {Post Literals, of height 1, only}

Where, $x' = {}^0x$, $x = {}^1x$, and $x'' = {}^2x$ only

But, this is only appropriate for Post Literals (of height 1) and not (GF(n)) variables, since there is no expansion with respect to shifts and terms with exponents, i.e. x^2 $(x')^2$, $(x'')^2$.

Equation 9.3.6:

$f(x_1, x_2,\ldots, x_n) = x_1{}'f(x_1 = 0, x_2, x_3, \ldots, x_n) \oplus x_1f(x_1 = 1, x_2, x_3,\ldots, x_n) \oplus x_1{}''f(x_1 = 2, x_2, x_3, \ldots, x_n)$

Shannon Expansion for GF(3) Post Literals of height 1 only.

Theorem 9.3.1:

Galois Field, Shannon Expansion for Post Logic, using Post Literals of height 1 only, ($GF_{S\text{-}Post}$)

$f = {}^{0}xf_0 \oplus {}^{1}xf_1 \oplus {}^{2}xf_2$ $\{GF_{S\text{-}Post}\}$

Proof: This Galois Field Shannon Expansion for Post Logic can be verified as follows. First, recall that ${}^{0}x$, ${}^{1}x$, and ${}^{2}x$ are Post Literals of height one. The terms can be examined as given:

${}^{0}x \times f_0 = 1 \times f_0 = f_0$

since, ${}^{0}x = 1$ for x=0

$= 0$ for x=1, 2

and

${}^{1}x \times f_1 = 1 \times f_1$

since, ${}^{1}x=1$ for x=1

${}^{1}x=0$ for x=0, 2

and

${}^{2}x \times f_2 = 1 \times f_2$

since, ${}^{2}x = 1$ for x=2

${}^{2}x = 0$ for x=0, 1

The other terms produce similar results. Hence the GF(3) Shannon function expansion becomes:

GF(3) Shannon $f(x) = {}^{0}xf_0 \oplus {}^{1}xf_1 \oplus {}^{2}xf_2$

The $GF_{S\text{-}Post}$ tree in GF(3), or ternary logic, is shown in Figure 9.3.15.

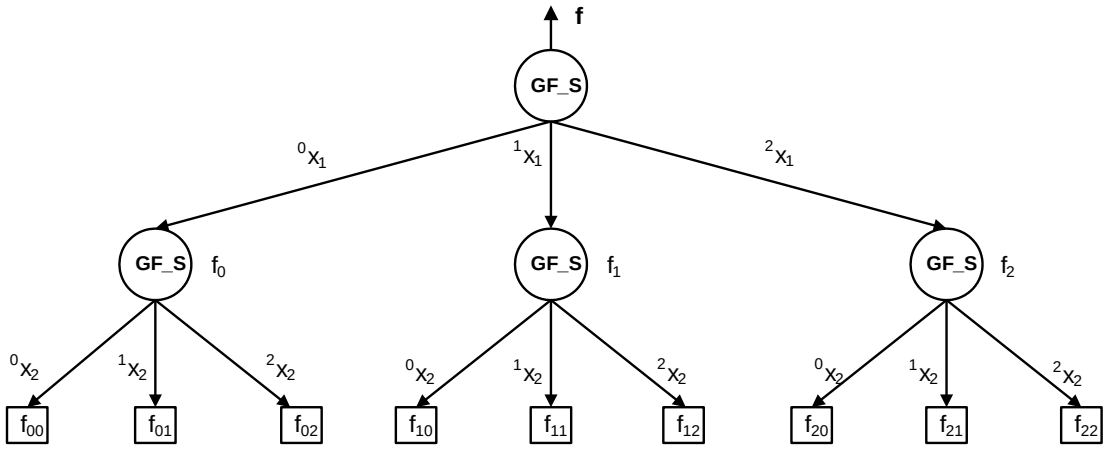


Figure 9.3.15 GF-Shannon Post Logic Tree for ternary logic.

Figure 9.3.16 presents the table of some ternary polynomials of single variable, i.e. functions of single variable Figure 9.3.16 to Figure 9.3.20. The application of the Shannon Expansion can be extended from the binary (GF(2)) to the Galois Field of size n, GF(n), case. This is done in a different manner than that of the trivial methodology presented for the Post Literal case above. Because Galois Theory links the ideas of group and field theory to that of algebra, in the derivation of the Shannon Expansion for GF(n), it is natural to first begin with irreducible polynomial forms. Although this concept has been alluded to previously, here the general algebraic form for a polynomial over a field is formally introduced.

Definition 9.3.2: A polynomial over a field F, has the algebraic form, $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2x^2 + a_1x^1 + a_0$ on an indeterminate x, where $a_n, a_{n-1}, \ldots, a_2, a_1, a_0 \in F$ and "+" denotes the GF(n) addition operation. The set of all polynomials over F is denoted as F[x]. In GF(3), for ternary logic, then, a general irreducible polynomial has the algebraic form,

$$f(x) = a \oplus bx \oplus cx^2$$

Where, x is a ternary variable with zero polarity a, b, c are constants, (0, 1, 2). This gives a total of $3\times3\times3=3^3=27$ polynomials (functions), some of which are given in Figure 9.3.16. Next, the Davio Expansions in a Galois Field, for the ternary logic case will be derived for the general polynomial. To determine the constants in terms of the cofactors, the algebraic function, $f = a \oplus bx \oplus cx^2$, must be evaluated for each of the three possible values of x. This is calculated as follows:

$$f_0 = f_{x=0} = a$$

$$f_1 = f_{x=1} = a \oplus b \oplus c$$

$$f_2 = f_{x=2} = a \oplus 2b \oplus c$$

Variable	Logic mapping			Operation's meaning or name
x	0	1	2	
$x+1=\bar{x}$	1	2	0	Shift operation
$x+2=\bar{\bar{x}}$	2	0	1	Shift operation
x^2	0	1	1	Power
x^2+1	1	2	2	
$x^2+2={}^0x$	2	0	0	Post literal
$2x$	0	2	1	(1 2) permutation
$2x+1$	1	0	2	
$2x+2$	2	1	0	Inverter
$2x^2$	0	2	2	
$2x^2+1={}^0x$	1	0	0	Reduced Post literal
$2x^2+2$	2	1	1	
$2x^2+x={}^2x$	0	0	1	Reduced Post literal

Figure 9.3.16. Examples of some polynomials in GF(3) and their meaning.

These functions must be solved simultaneously, utilizing only the mathematical operators within the algebra, specifically, GF(3) addition and multiplication. Since the constant a is already determined, no further work must be done for its specification. Hence, only

constants b and c must be derived. The constants for the algebraic function, $f = a \oplus bx \oplus cx^2$, are given as shown below.

$$a = f_0$$

$$b = 2f_1 \oplus f_2$$

$$c = 2(f_0 \oplus f_1 \oplus f_2)$$

Substituting these constants back into the algebraic function, the formula for the GF(3) Davio-0 expansion, denoted as $GF(3)_{D\text{-}0}$ is given in Equation 9.3.6. Expansion trees provide a graphical representation of functional components. As a diagram of cofactors and multipliers (constants), they provide a visual depiction of decision trees, which are a useful tool in deriving the forms of an algebraic family. The tree for the GF(3) Davio-0 Expansion is given in Figure 9.3.19.

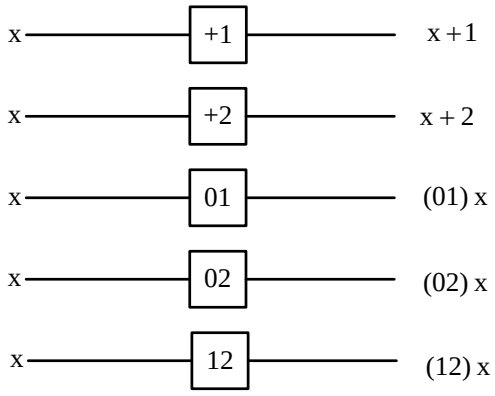


Figure 9.3.17. Realization of all useful reversible ternary functions of a single variable. This figure introduces also the notations.

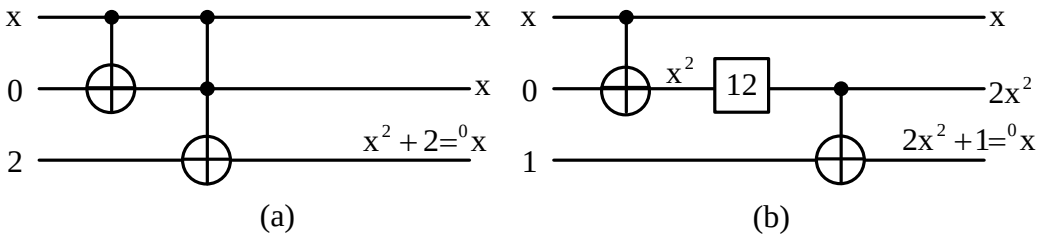


Figure 9.3.18. Realization of Post literals and Reduced Post literals (a) Post literal ${}^0x$, (b) reduced Post literal. As we see, ancilla bit is necessary in both cases.

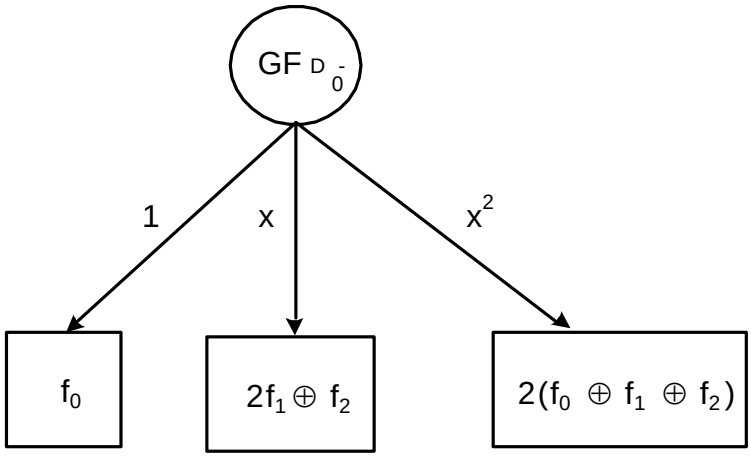


Figure 9.3.19. GF(3) Davio-0 Expansion Tree.

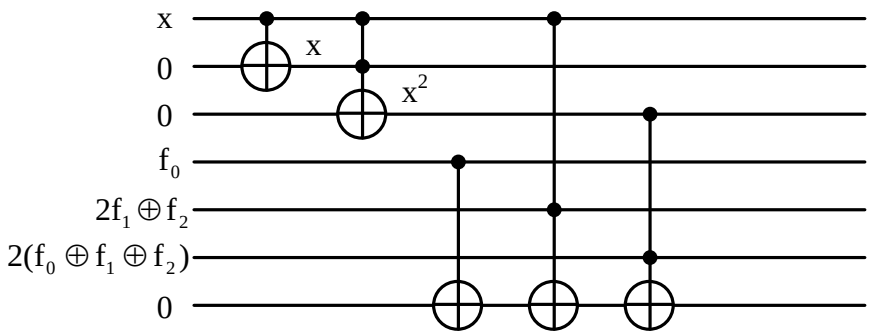


Figure 9.3.20. Ternary quantum array that realizes the GF(3)Davio-0 expansion from Figure 9.3.18.

The Davio-1 Expansion is found by defining x in terms of x'. This can be done with the substitution of $x = x' \oplus 2$, since the operations are for a ternary logic, in GF(3). This is given as follows.

$f = a \oplus bx \oplus cx^2$

$f = f_0 \oplus (2f_1 \oplus f_2)x \oplus 2(f_0 \oplus f_1 \oplus f_2)x^2$

$f = f_0 \oplus (2f_1 \oplus f_2)(x' \oplus 2) \oplus 2(f_0 \oplus f_1 \oplus f_2)(x' \oplus 2)(x' \oplus 2)$

$f = f_0 \oplus 2f_1x' \oplus f_1 \oplus f_2x' \oplus 2f_2 \oplus 2(f_0 \oplus f_1 \oplus f_2)(x'^2 \oplus 2x' \oplus 2x' \oplus 1)$

$f = f_0 \oplus 2f_1x' \oplus f_1 \oplus f_2x' \oplus 2f_2 \oplus 2(f_0 \oplus f_1 \oplus f_2)(x'^2 \oplus x' \oplus 1)$

$f = f_0 \oplus 2f_1x' \oplus f_1 \oplus f_2x' \oplus 2f_2 \oplus 2(f_0x'^2 \oplus f_2x' \oplus f_0 \oplus f_1x'^2 \oplus f_1x' \oplus f_1 \oplus f_2x'^2 \oplus f_2x' \oplus f_2)$

$f = f_0 \oplus 2f_1x' \oplus f_1 \oplus f_2x' \oplus 2f_2 \oplus 2f_0x'^2 \oplus 2f_0x' \oplus 2f_0 \oplus 2f_1x'^2 \oplus 2f_1x' \oplus 2f_1 \oplus 2f_2x'^2 \oplus 2f_2x' \oplus 2f_2$

$f = (f_0 \oplus f_1 \oplus 2f_2 \oplus 2f_0 \oplus 2f_1 \oplus 2f_2) \oplus (2f_1 \oplus f_2 \oplus 2f_0 \oplus 2f_1 \oplus 2f_2)x' \oplus (2f_0 \oplus 2f_1 \oplus 2f_2)x'^2$

The expansion is denoted as $GF(3)_{D\text{-}1}$ and simplified to the form given in Equation 9.3.8. The tree for this expansion is given in Figure 9.3.21.

Equation 9.3.8: GF(3) Davio-1, $f = f_2 \oplus (2f_0 \oplus f_1)x' \oplus 2\,(f_0 \oplus f_1 \oplus f_2)(x')^2$

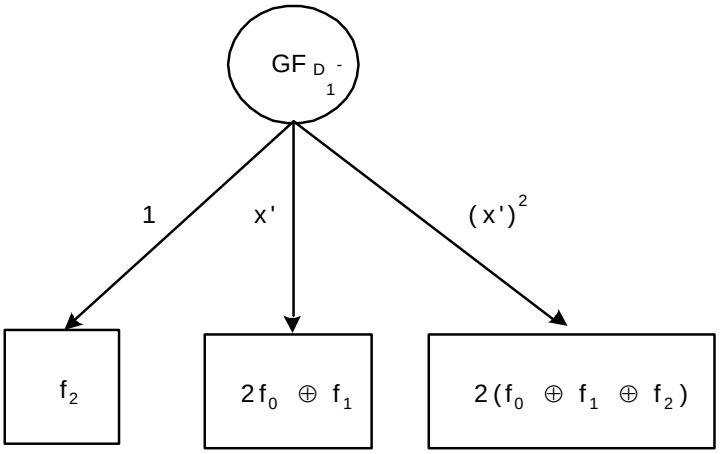


Figure 9.3.21. GF(3) Davio-1 Expansion Tree.

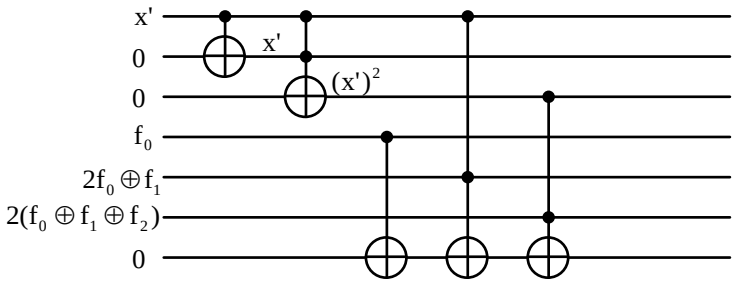


Figure 9.3.22. Ternary quantum array that realizes the GF(3) Davio-1 expansions from Figure 9.3.20.

The Davio-2 expansion is obtained similarly. The x variable is defined in terms of x'', with the substitution $x = x'' \oplus 1$.

$f = a \oplus bx \oplus cx^2$

$f = f_0 \oplus (2f_1 \oplus f_2)x \oplus 2(f_0 \oplus f_1 \oplus f_2)x^2$

$f = f_0 \oplus (2f_1 \oplus f_2)(x'' \oplus 1) \oplus 2(f_0 \oplus f_1 \oplus f_2)(x'' \oplus 1)(x'' \oplus 1)$

$f = f_0 \oplus (2f_1x'' \oplus 2f_1 \oplus f_2x'' \oplus f_2) \oplus 2(f_0 \oplus f_1 \oplus f_2)(x''^2 \oplus 2x'' \oplus 1)$

$f = f_0 \oplus 2f_1x'' \oplus 2f_1 \oplus f_2x'' \oplus f_2 \oplus 2(f_0x''^2 \oplus 2f_0x'' \oplus f_0 \oplus f_1x''^2 \oplus 2f_1x'' \oplus f_1 \oplus f_2x''^2 \oplus 2f_2x'' \oplus f_2)$

$f = f_0 \oplus 2f_1x'' \oplus 2f_1 \oplus f_2x'' \oplus f_2 \oplus 2f_0x''^2 \oplus f_0x'' \oplus 2f_0 \oplus 2f_1x''^2 \oplus f_1x'' \oplus 2f_1 \oplus 2f_2x''^2 \oplus f_2x'' \oplus f_2$

$f = (f_0 \oplus 2f_1 \oplus f_2 \oplus 2f_0 \oplus 2f_1 \oplus 2f_2) \oplus (2f_1 \oplus f_2 \oplus f_0 \oplus f_1 \oplus f_2)x'' \oplus (2f_0 \oplus 2f_1 \oplus 2f_2)x''^2$

The Davio-2 expansion, denoted as $GF(3)_{D\text{-}2}$, simplifies to the form given in Equation 9.3.11. The tree for this expansion is given in Figure 9.3.23.

Equation 9.3.9: GF(3) Davio-2, $f = f_1 \oplus (f_0 \oplus 2f_2)x'' \oplus 2(f_0 \oplus f_1 \oplus f_2)(x'')^2$

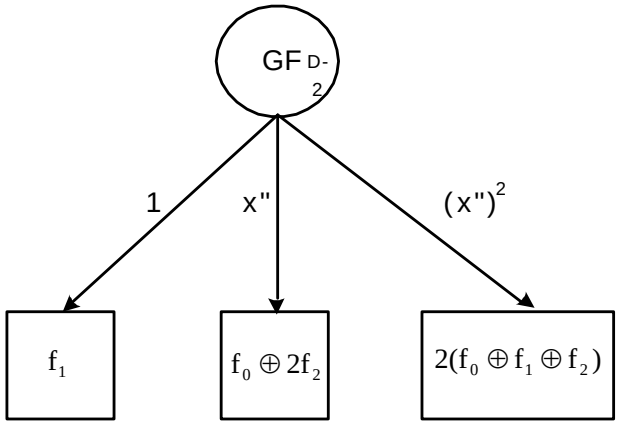


Figure 9.3.23. Expansion tree for GF(3) Davio-2.

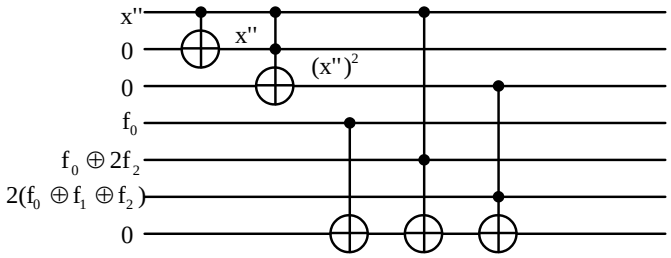


Figure 9.3.24. Ternary quantum array that realizes the GF(3) Davio-2 expansion from Figure 9.3.23.

For the development of an algebraic family of forms, expansions of the original function are performed. For a Galois Field of size n, only n+1 expansion types are possible for the derivation of all forms. For GF(3), these are the Shannon, Davio-0, Davio-1, and Davio-2, denoted as $GF(3)_{D\text{-}0}$, $GF(3)_{D\text{-}1}$, and $GF(3)_{D\text{-}2}$ respectively. In GF(3), the functions defining the Davio family of universal literals are shown in Figure 9.3.25 below.

Davio Family

1	x	x^2	Davio-0
1	x’	$(x')^2$	Davio-1
1	x’’	$(x'')^2$	Davio-2

Figure 9.3.25. Comparison of ternary GF Davio expansions.

To verify that new expansions are valid and applicable, the functions defining the branches of the expansion trees must be proven linearly independent. First, the definition of linear independence is given.

Definition 9.3.3: A vector space with $V = \{v_1, \ldots, v_n\}$ as a set of vectors is given. The set of vectors is linearly independent if the equation:

$$c_1v_1 + \ldots + c_nv_n = 0$$

holds, if and only if all of the multipliers $c_1 = \ldots = c_n = 0$. Otherwise, a vector set that is not linearly independent is linearly dependent [Perkowski97b]. The expansions such as

the Shannon, Positive and Negative Davio can be applied to functions, as a variable separation technique, creating an expansion tree diagram. In expansion tree diagrams, several expansions may be combined, such that each node on a level, corresponding to an expansion variable, has one of the defined expansions. The total function (over the entire tree), in its new form, can then be re-constructed by combining the cofactors and multipliers for each of the branches with the EXOR operation.

9.3.4. Ternary Pseudo-Kronecker Expansions.

To better understand multi-level expansion trees, an example shown in Figure 9.3.26 is given of a Pseudo-Kronecker Expansion (any subset of expansions, out of all possible expansions, is chosen for every level of the ordered tree) for a three variable function f(a, b, c) in GF(3). (The GF-Pseudo-Kronecker Expansion is defined in Definition 9.3.19.) This figure is constructed purely from the single expansion trees given previously, connected together. The first level expansions are with respect to variable a, the second level expansions are with respect to variable b, and the third level expansions are with respect to variable c. The cofactors corresponding to previous expansions are not shown, but assumed, due to the existence of the subsequent expansion. The end nodes denote the final cofactors. The notation in the subscript of these cofactors refers to the particular expansions originating the branch, i.e. f_{012} refers to the fact that an expansion with respect to a = 0, then an expansion with respect to b = 1, and another expansion with respect c = 2, were previously performed on the branch.

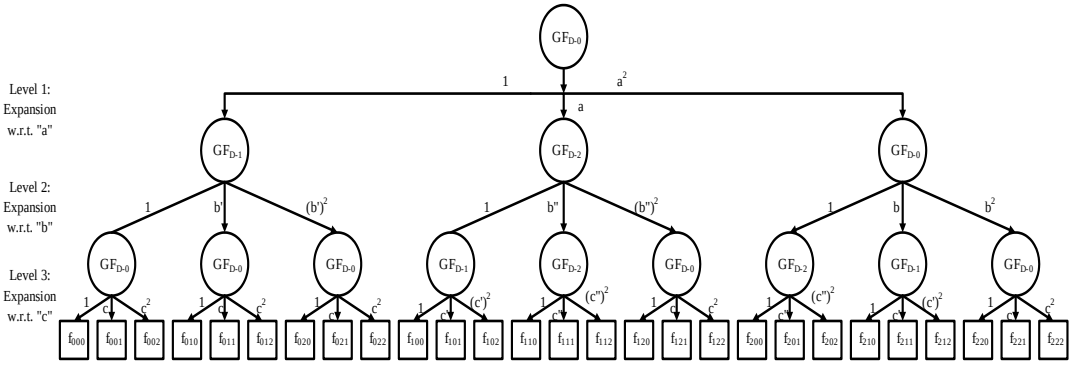


Figure 9.3.26. GF(3)-Pseudo-Kronecker Expansions.

For the arbitrary expansions depicted in Figure 9.3.26 the total function can be constructed in terms of each of the branches describing the multipliers and cofactors. Starting at the bottom of the tree, for each level the terms are combined together by an EXOR operation, then collectively applied to the previous expansion, as this expression acts as a cofactor for the multiplier in the next higher level. The building of the function shown in the decision tree of Figure 9.3.26 is demonstrated as follows:

<u>Level 3:</u>

For the (left branch) three GF_{D-0} Expansions, the terms are:

$f_{000}(1) \oplus f_{001}(c) \oplus f_{002}(c^2)$

$f_{010}(1) \oplus f_{011}(c) \oplus f_{012}(c^2)$

$f_{020}(1) \oplus f_{021}(c) \oplus f_{022}(c^2)$

For the (middle branch) GF_{D-1}, GF_{D-2}, and GF_{D-0} Expansions, the terms are as follows, respectively:

$f_{100}(1) \oplus f_{101}(c') \oplus f_{102}(c')^2$

$f_{110}(1) \oplus f_{111}(c'') \oplus f_{112}(c'')^2$

$f_{120}(1) \oplus f_{121}(c) \oplus f_{122}(c^2)$

For the (right branch) GF_{D-2}, GF_{D-1}, and GF_{D-0} Expansion, the terms are as follows, respectively:

$f_{200}(1) \oplus f_{201}(c'') \oplus f_{202}(c'')^2$

$f_{210}(1) \oplus f_{211}(c') \oplus f_{212}(c')^2$

$f_{220}(1) \oplus f_{221}(c) \oplus f_{222}(c^2)$

Level 2:

For the (left branch) GF_{D-1} Expansion, the term is:

$\{(f_{000}(1) \oplus f_{001}(c) \oplus f_{002}(c^2))(1) \oplus (f_{010}(1) \oplus f_{011}(c) \oplus f_{012}(c^2))(b') \oplus (f_{020}(1) \oplus f_{021}(c) \oplus f_{022}(c^2))(b')^2\}$

For the (middle branch) GF_{D-2} Expansion, the term is:

$\{(f_{100}(1) \oplus f_{101}(c') \oplus f_{102}(c')^2)(1) \oplus (f_{110}(1) \oplus f_{111}(c'') \oplus f_{112}(c'')^2)(b'') \oplus (f_{120}(1) \oplus f_{121}(c) \oplus f_{122}(c^2))(b'')^2\}$

For the (right branch) GF_{D-0} Expansion, the term is:

$\{(f_{200}(1) \oplus f_{201}(c'') \oplus f_{202}(c'')^2)(1) \oplus (f_{210}(1) \oplus f_{211}(c') \oplus f_{212}(c')^2)(b) \oplus (f_{220}(1) \oplus f_{221}(c) \oplus f_{222}(c^2))(b)^2\}$

These terms are then used as cofactors for level 1.

Level 1:

For the (left branch) of the GF_{D-0} Expansion, the term is:

$\{(f_{000}(1) \oplus f_{001}(c) \oplus f_{002}(c^2))(1) \oplus (f_{010}(1) \oplus f_{011}(c) \oplus f_{012}(c^2))(b') \oplus (f_{020}(1) \oplus f_{021}(c) \oplus f_{022}(c^2))(b')^2\}(1)$

For the (middle branch) of the GF_{D-0} Expansion, the term is:

$\{(f_{100}(1) \oplus f_{101}(c') \oplus f_{102}(c')^2)(1) \oplus (f_{110}(1) \oplus f_{111}(c'') \oplus f_{112}(c'')^2)(b'') \oplus (f_{120}(1) \oplus f_{121}(c) \oplus f_{122}(c^2))(b'')^2\}(a)$

For the (right branch) of the $GF_{D\text{-}0}$ Expansion, the term is:

$\{(f_{200}(1) \oplus f_{201}(c'') \oplus f_{202}(c'')^2)(1) \oplus (f_{210}(1) \oplus f_{211}(c') \oplus f_{212}(c')^2)(b) \oplus (f_{220}(1) \oplus f_{221}(c) \oplus f_{222}(c^2))(b)^2\}(a)^2$

Constructing the total function, the branches are simply combined with the EXOR operation, as follows.

$f(a,b,c) = [\{(f_{000}(1) \oplus f_{001}(c) \oplus f_{002}(c^2))(1) \oplus (f_{010}(1) \oplus f_{011}(c) \oplus f_{012}(c^2))(b') \oplus (f_{020}(1) \oplus f_{021}(c) \oplus f_{022}(c^2))(b')^2\}(1)] \oplus [\{(f_{100}(1) \oplus f_{101}(c') \oplus f_{102}(c')^2)(1) \oplus (f_{110}(1) \oplus f_{111}(c'') \oplus f_{112}(c'')^2)(b'') \oplus (f_{120}(1) \oplus f_{121}(c) \oplus f_{122}(c^2))(b'')^2\}(a)] \oplus [\{(f_{200}(1) \oplus f_{201}(c'') \oplus f_{202}(c'')^2)(1) \oplus (f_{210}(1) \oplus f_{211}(c') \oplus f_{212}(c')^2)(b) \oplus (f_{220}(1) \oplus f_{221}(c) \oplus f_{222}(c^2))(b)^2\}(a)^2]$

Example 9.3.12: Given the function described by the K-map given in Figure 9.3.27, apply the GF(3) expansions of Figure 9.3.26 and flatten the tree into a function expressed algebraically.

ab \ c	0	1	2
00	0	2	0
01	1	2	2
02	2	2	0
10	1	0	0
11	1	0	1
12	2	2	1
20	0	1	0
21	0	2	1
22	1	1	0

Figure 9.3.27. A map of a ternary function of three variables.

Recall that for GF(3), the Davio Expansions are as follows:

$GF_{D\text{-}0}$: $f = f_0 \oplus (2f_1 \oplus f_2)x \oplus 2(f_0 \oplus f_1 \oplus f_2)x^2$

$GF_{D\text{-}1}$: $f = f_2 \oplus (2f_0 \oplus f_1)x' \oplus 2(f_0 \oplus f_1 \oplus f_2)(x')^2$

$GF_{D\text{-}2}$: $f = f_1 \oplus (f_0 \oplus 2f_2)x'' \oplus 2(f_0 \oplus f_1 \oplus f_2)(x'')^2$

From the K-map and the Davio Expansions listed above, the corresponding cofactor for each node of Figure 9.3.30 is determined. For convenience, these nodes are numerically labeled (shown in circles) in Figure 9.3.28.

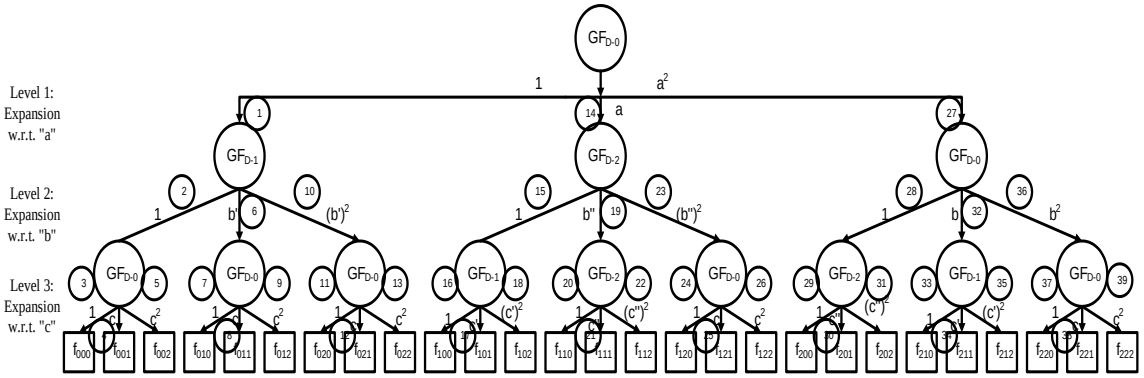


Figure 9.3.28. Complete ternary tree that uses Galois Field Davio Expansions. It expands the ternary function from Figure 9.3.27.

The cofactors for each node in Figure 9.3.28 are determined from the K-map given in Figure 9.3.27. These calculations are given as follows.

<u>Node 1:</u>

$$f_{a=0} = \begin{bmatrix} 020 \\ 122 \\ 220 \end{bmatrix}$$

<u>Node 2:</u>

$f_{ab=02} = [220]$

<u>Node 3:</u>

$f_{abc=020} = 2$

Node 4:

$(2f_1 + f_2)|f_{ab=02} = 2(2) + 0$

$= 1$

Node 5:

$\{2(f_0 + f_1 + f_2)\}|f_{ab=02} = 2(2 + 2 + 0)$

$= 2$

Node 6:

$(2f_0 + f_1)|f_{a=0,b=0,1} = 2[020] + [122]$

$= [010] + [122]$

$= [102]$

Node 7:

$f_0|_{[102]} = 1$

Node 8:

$2f_1 + f_2|_{[102]} = 2(0) + 2$

$= 2$

Node 9:

$2(f_0 + f_1 + f_2)|_{[102]} = 2(1 + 0 + 2)$

$= 0$

Node 10:

$2(f_0 + f_1 + f_2)|f_{a=0,b} = 2\{[020] + [122] + [220]\}$

$= 2[002]$

$= [001]$

Node 11:

$f_0|_{[001]} = 0$

Node 12:

$(2f_1 + f_2)|_{[001]} = 2(0) + 1$

$= 1$

Node 13:

$\{2(f_0 + f_1 + f_2)\}|_{[001]} = 2(0 + 0 + 1)$

$= 2$

Node 14:

$$(2f_1 + f_2)|_{a=1,2} = 2\begin{bmatrix}100\\101\\221\end{bmatrix} + \begin{bmatrix}010\\021\\110\end{bmatrix}$$

$$= \begin{bmatrix}200\\202\\112\end{bmatrix} + \begin{bmatrix}010\\021\\110\end{bmatrix}$$

$$= \begin{bmatrix}210\\220\\222\end{bmatrix}$$

Node 15:

$f_{b=1} = [220]$

Node 16:

$f_2|_{[220]} = 0$

Node 17:

$(2f_0 + f_1)|_{[220]} = 0$

Node 18:

$2(f_0 + f_1 + f_2)|_{[220]} = 2(2 + 2 + 0)$

$= 2$

Node 19:

$f_0 + 2f_2 = [021]$

Node 20:

$f_1|[021] = 2$

Node 21:

$(f_0 + 2f_2)|_{[021]} = 0 + 2(1)$

$= 2$

Node 22:

$2(f_0 + f_1 + f_2)|_{[021]} = 2(0 + 2 + 1)$

$= 2(0)$

$= 0$

Node 23:

$2(f_0 + f_1 + f_2) = 2\{[210] + [220] + [222]\}$

$= 2[022]$

$= [011]$

Node 24:

$f_0|_{[011]} = 0$

Node 25:

$(2f_1 + f_2)|_{[011]} = 2(1) + 1$

$= 0$

Node 26:

$2(f_0 + f_1 + f_2)|_{[011]} = 2(0 + 1 + 1)$

$= 1$

Node 27:

$$2(f_0 + f_1 + f_2) = 2\{\begin{bmatrix}020\\122\\220\end{bmatrix} + \begin{bmatrix}100\\101\\221\end{bmatrix} + \begin{bmatrix}010\\021\\110\end{bmatrix}\}$$

$$= 2\begin{bmatrix}100\\211\\221\end{bmatrix}$$

$$= \begin{bmatrix}200\\122\\112\end{bmatrix}$$

Node 28:

$f_0 = [200]$

Node 29:

$f_1|_{[200]} = 0$

Node 30:

$(f_0 + 2f_2)|_{[200]} = 2 + 2(0)$

$= 2$

Node 31:

$\{2(f_0 + f_1 + f_2)\}|_{[200]} = 2\{2 + 0 + 0\}$

$= 1$

Node 32:

$2f_1 + f_2 = 2[122] + [112]$

$= [211] + [112]$

$= [020]$

Node 33:

$f_2|_{[020]} = 0$

Node 34:

$(2f_0 + f_1)|_{[020]} = 2(0) + 2$

$= 2$

Node 35:

$2(f_0 + f_1 + f_2)|_{[020]} = 2(0 + 2 + 0)$

$= 1$

Node 36:

$2(f_0 + f_1 + f_2) = 2\{[200] + [122] + [112]\}$

$= 2[101]$

$= [202]$

Node 37:

$f_0|_{[202]} = 2$

Node 38:

$(2f_1 + f_2)|_{[202]} = 2(0) + 2$

$= 2$

Node 39:

$\{2(f_0 + f_1 + f_2)\}|_{[202]} = 2(2 + 0 + 2)$

= 2

From these expansions, the root terms for Figures 9.3.30 and 9.3.32 are determined. This is shown in Figure 9.3.33.

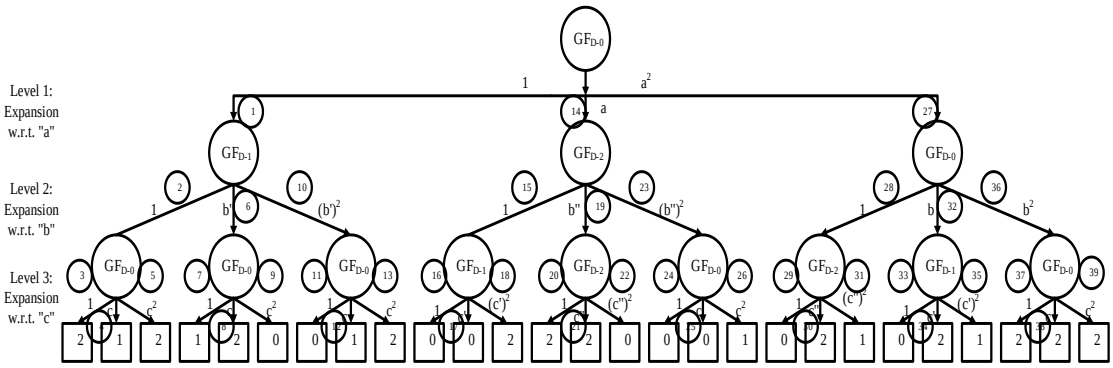


Figure 9.3.29. Another ternary tree

From the completed decision tree in Figure 9.3.29, an algebraic expression can be written. This is given and simplified as follows.

$$f = \{(2 \oplus c \oplus 2c^2) \oplus (1 \oplus 2c)b' \oplus (c \oplus 2c^2)b'^2\} \oplus$$

$$\{2c'^2 \oplus (2 \oplus 2c'')b'' \oplus c^2(b'')^2\}a \oplus$$

$$\{(2c'' \oplus c''^2) \oplus (2c' \oplus c'^2)b \oplus (2 \oplus 2c \oplus 2c^2)b^2\}a^2$$

$$= \{2 \oplus c \oplus 2c^2 \oplus b' \oplus 2b'c \oplus b'^2c \oplus 2b'^2c^2\}$$

$$\{2c'^2 \oplus 2b'' \oplus 2b''c'' \oplus b''^2c^2\}a \oplus$$

$$\{2c'' \oplus c''^2 \oplus 2bc' \oplus bc'^2 \oplus 2b^2 \oplus 2b^2c \oplus 2b^2c^2\}a^2$$

$$= 2 \oplus c \oplus 2c^2 \oplus b' \oplus 2b'c \oplus b'^2c \oplus 2b'^2c^2 \oplus 2ac'^2 \oplus 2ab'' \oplus 2ab''c'' \oplus ab''^2c^2$$

$$\oplus 2a^2c'' \oplus a^2c''^2 \oplus 2a^2bc' \oplus a^2bc'^2 \oplus 2a^2b^2 \oplus 2a^2b^2c \oplus 2a^2b^2c^2$$

The algebraic expression for the function can be verified by calculating the output for given values of the variables a, b, and c, and comparing these values with those listed in the K-map in Figure 9.3.31. Only a few verifications are given here.

For abc = 000:

f_{000} = 2 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕

0

= 0

For abc = 111:

f_{111} = 2 ⊕ 1 ⊕ $2(1)^2$ ⊕ 2 ⊕ 2(2)(1) ⊕ $(2)^2(1)$ ⊕ $2(2)^2(1)^2$ ⊕ $2(1)(2)^2$ ⊕ 2(1)(0) ⊕ 2(1)(0)(0) ⊕ $(1)(0)^2(1)^2$ ⊕ $2(1)^2(0)$ ⊕ $(1)^2(0)^2$ ⊕ $2(1)^2(1)(2)$ ⊕ $(1)^2(1)(2)^2$ ⊕ $2(1)^2(1)^2$ ⊕ $2(1)^2(1)^2(1)$ ⊕ $2(1)^2(1)^2(1)^2$

= 2 ⊕ 1 ⊕ 2 ⊕ 2 ⊕ 1 ⊕ 1 ⊕ 2 ⊕ 2 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 2 ⊕ 2 ⊕ 2

= 0

For abc = 222:

f_{222} = 2 ⊕ 2 ⊕ 2(2)(2) ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 0 ⊕ 2(2) ⊕ 2(2)(1) ⊕ 2(2)(1)(1) ⊕ $2(1^2)(1^2)$ ⊕ (2)(2)(2) ⊕ 2(2)(2)(1) ⊕ $2(2)(1)^2$ ⊕ 2(2)(2)(2)(0) ⊕ 2(2)(2)(0) ⊕ (2)(2)(2)(2)(2) ⊕ (2)(2)(2)(2)(2)(2) ⊕ (2)(2)(2)(2)(2)(2)(2)

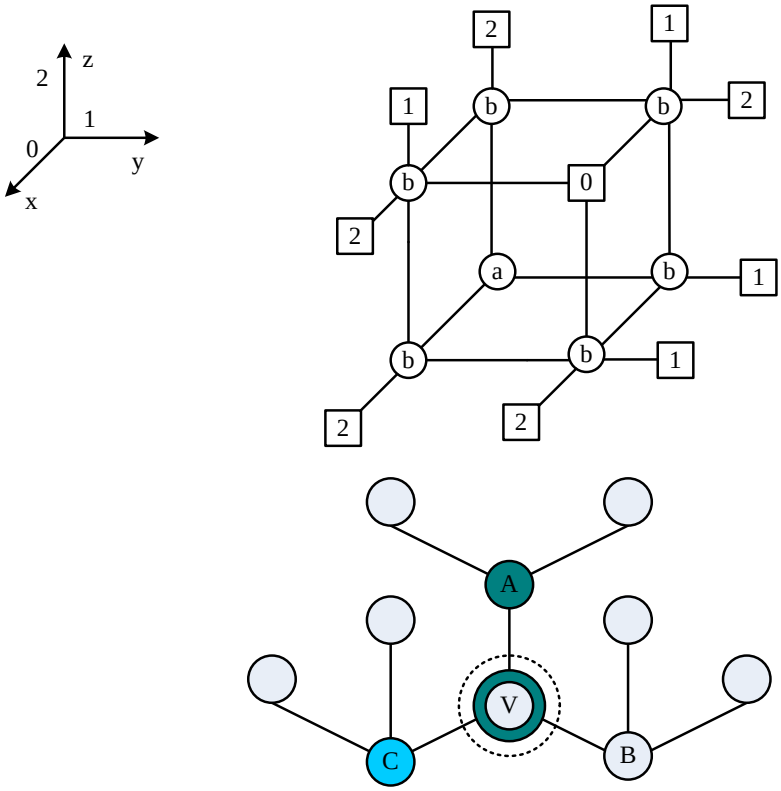
= 2 ⊕ 2 ⊕ 2 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 2 ⊕ 2 ⊕ 2 ⊕ 1 ⊕ 2 ⊕ 1 ⊕ 2

= 0

9.3.5. Ternary Lattices.

The same as for binary lattices, the ternary lattice can be created either by expansions of symmetric function decision diagrams or by our approach based on forward and reverse

expansions. Figure 9.3.31 presents the three dimensional expansion of a Lattice Ternary Diagram (a Regular Diagram but the regularity is seen only in three dimensional geometrical space).



a \ b	0	1	2
0	2	1	1
1	2	1	1
2	2	2	2

$$F = 2\,{}^{0}a\,{}^{0}b + {}^{0}a\,{}^{1}b + {}^{0}a\,{}^{2}b + 2\,{}^{1}a\,{}^{0}b + {}^{1}a\,{}^{1}b + {}^{1}a\,{}^{2}b + 2\,{}^{2}a\,{}^{0}b + 2\,{}^{2}a\,{}^{1}b + 2\,{}^{2}a\,{}^{2}b$$

Figure 9.3.31. Explanation of the placement of a ternary Shannon Expansion in three dimensional space. Of course, this types of expansions and their 3D layouts can be done for any type of Shannon expansions for logic of radix 3.

9.4. The GF-Kronecker type Expansions.

The GF Hierarchical definitions, forms, decision diagrams, and trees for ternary logic are as follows.

Definition 9.4.1: The GF-Kronecker Expansion (GF-KRO-EXP) is obtained if only one expansion, out of all possible expansions (for GF(3) GF_S, GF_{D-0}, GF_{D-1}, GF_{D-2}), is chosen, for every level of an ordered tree. Figure 9.4.1 shows an example of a GF-KRO Tree.

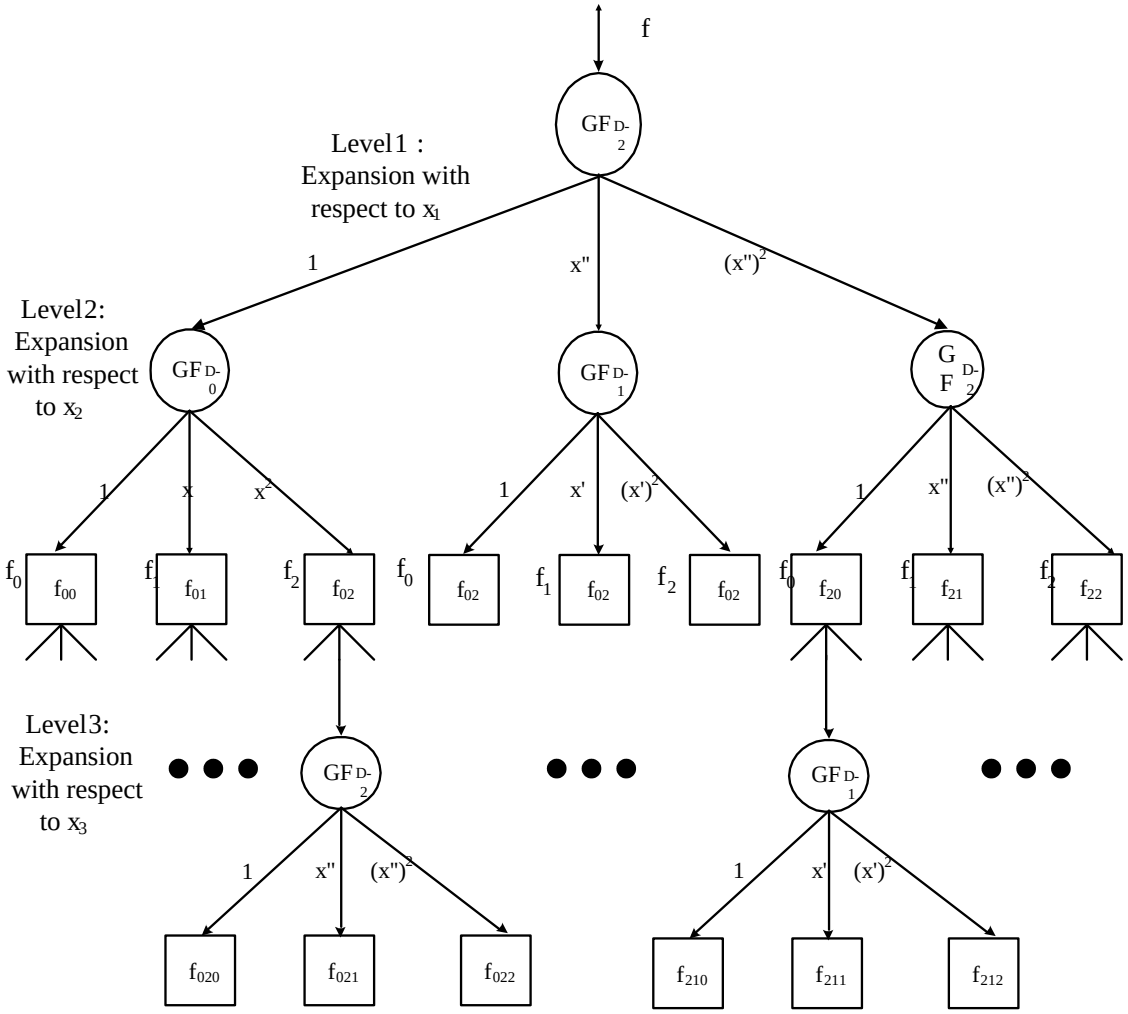


Figure 9.4.1: GF-KRO for GF(3)

Definition 9.4.2: The GF-Pseudo-Kronecker Expansion is obtained if any subset of expansions, out of all possible expansions, is chosen for every level of an ordered tree. Figure 9.4.2 shows an example of a GF-PKRO Tree.

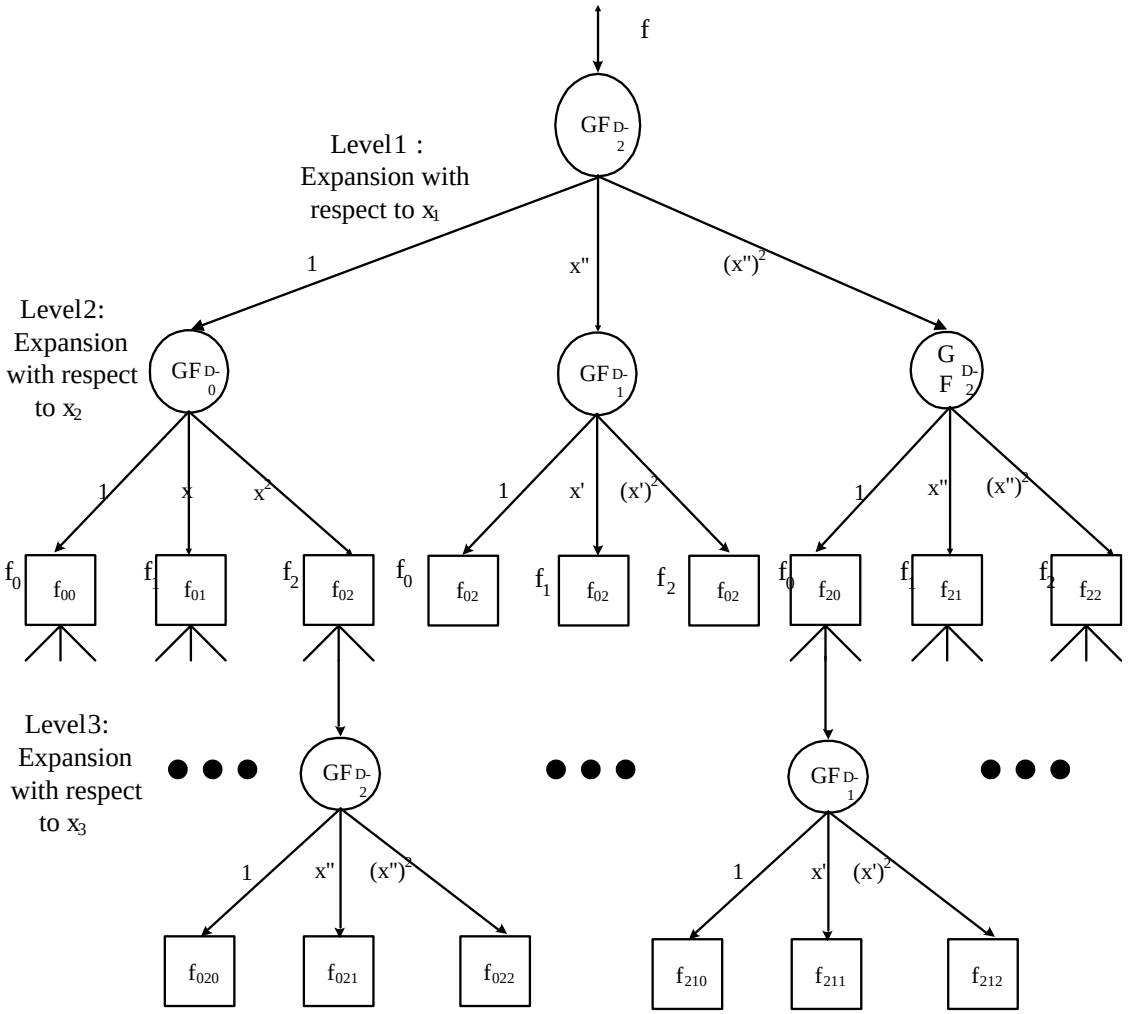


Figure 9.4.2: GF-PKRO for GF(3).

Definition 9.4.3: The GF-Free Expansion is obtained if the expansion variable and type can be freely chosen, with no ordering of variables for each and every level of the tree. An example is shown in Figure 9.4.3.

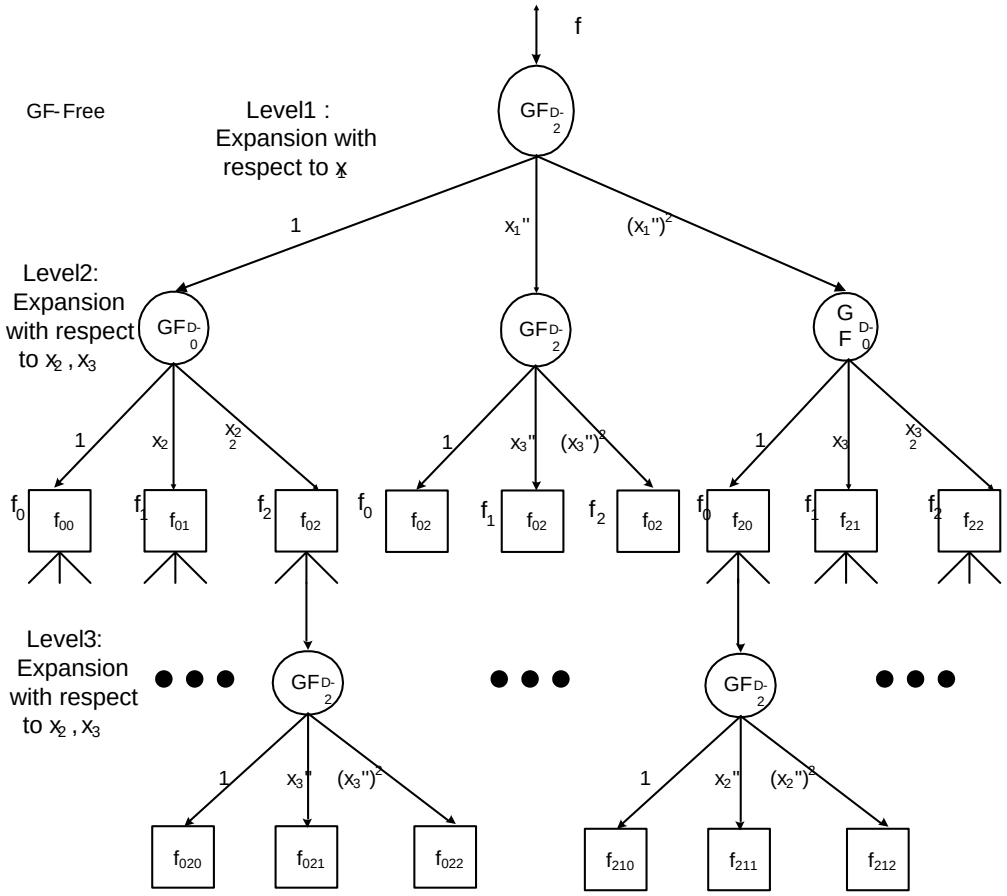


Figure 9.4.3: GF-Free for GF(3).

So far we showed in this chapter how to create some types of ternary trees based on polynomial expansions. As we know, from ternary trees one can create lattices and diagrams the same way as for binary logic, so I will not write here much about these issues. Examples of creating diagrams and lattices for ternary and quaternary circuits can be easily created based on the above presentation.

9.5. The GF Hierarchy.

The Galois Field Hierarchy for GF(3) is presented in Figure 9.5.1. In this table since GF(3) is considered, the possible Davio Expansions are with respect to the powers of x, x', and x''. Note that if GF(4) were to be considered, the possible expansions would be

with respect to the powers of x, x', x'', and x''', all possible logic states for the quaternary GF logic.

Expansion	**Tree**	**Decision Diagram**	**Forms**
GFS	GFS Tree	GF-TDD	Canonical GF-SOP
GFD-0 GFD-1 GFD-2	GFD-0 Tree GFD-1 Tree GFD-2 Tree	GF-FDD GF-FDD GF-FDD	SPRM (Single Polarity GF-RM)
GF-KRO any single expansion from set (GFS, GFD-0, GFD-1, GFD-2) per level and with ordered variables	GF-KRO Tree	GF-KDD	GF-KRM
GF-PSKRO (GFS, GFD-0, (GFD-1, GFD-2) any subset per level, with ordered variables	GF-Pseudo Kronecker Tree	GF-PKDD	GF-PKRM
GF-FKRO Any subset (GFS, GFD-0, GFD-1, GFD-2) no order of variables	Free GF-KRO Tree	GF-FKDD	GF-FKRM

Figure 9.5.1: The Galois Field Logic Hierarchy.

The hierarchy introduced above systematizes various multiple-valued concepts that are related to all circuits in which the combining operator has a property of a group. Some of these structures were known to other authors, but several of their publications appeared after I already derived all above expansions and structures. Also, it is for the first time

that it was shown here that all these concepts apply to reversible and quantum logic and not only to classical multiple-valued logic.

9.6. Quantum Circuits based on Galois Fields.

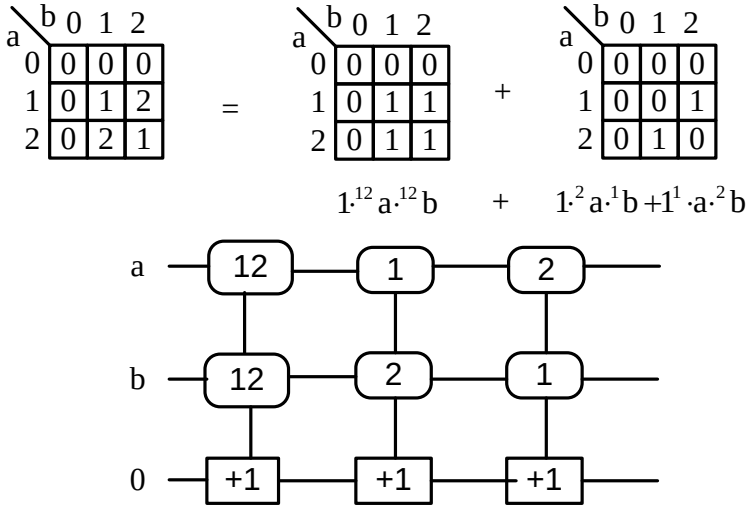


Figure 9.6.1. Realization of Ternary multiplication for Galois Field (3) logic. This is modulo multiplication for prime number.

The ternary Modulo Multiplication (and also GF(3) multiplication) is presented in Figure 9.6.1. Observe an interesting fact. In binary we have one GF(2) logic which is AND-EXOR logic. Realization of GF(4) addition is shown in Figure 9.6.3. The cost is high and can be compared with the respective vector binary circuit from section 9.7. These examples illustrate that there are many ways to synthesize regular MV quantum circuits only experimental results can help determine the base of gates, if such a base exists. This is not done in my dissertation but I have already outlined how it can be done in principle.

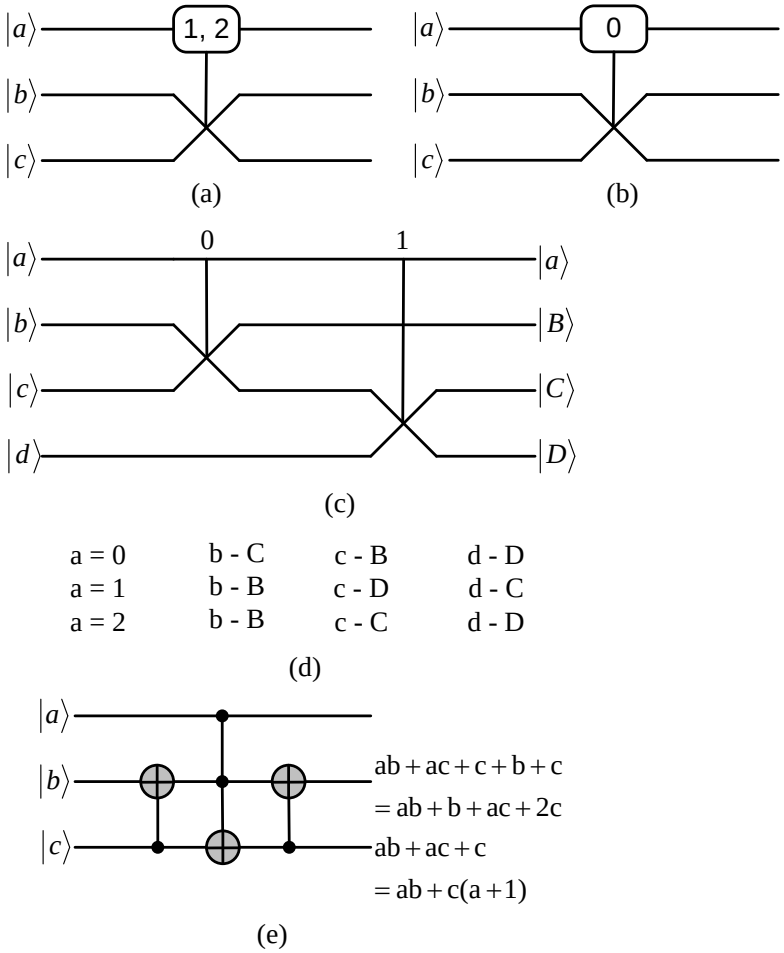


Figure 9.6.4. Synthesis of Ternary Fredkin gate by using analogy to binary case and Galois Field (3) algebraic transformations. (a) Fredkin-like ternary gate proposal controlled by value of Generalized Post Literal $^{1,2}|a\rangle$, (b) Fredkin-like proposal ternary gate proposal controlled by Post literal $^{0}|a\rangle$, (c) Fredkin-like gate proposal with MS type controls, (d) calculation of swaps for various values of controls, (e) another type of Fredkin generalization to ternary. These types of gates switch between various affine functions of two variables and generalize Dipal Gate family from binary to ternary. Symbol + is modulo addition.

Now we will show design on the level of ternary gates. Figure 9.6.4 presents the Ternary Fredkin gate. As we can check, for every value of variable the functions in lower two qutrits are affine functions (as realized with ternary Feynman gates). The same can be verified for the original Dipal Gate and next for all gates from Dipal Gate Family.

9.7. MVL Orthogonal and Linearly Independent Expansions.

As we know, the Linearly Independent (Orthogonal as special case) expansions for binary case use EXOR gate, and are generalizations of the Davio expansions. There is therefore a strong link between polynomial and Linearly Independent logics. For finite multiple-valued logic these expansions are based on Galois Field Addition gate or Modulo Addition, and in general, for arbitrary algebras, they should have at least one linear (group) operation as addition. In particular, they include S (Shannon Expansion). Actually, Shannon did not invent the case for N>2 and I call it Shannon Expansion for historical reasons and by analogy.

Observe that the OR operator in such generalized Shannon expansions can be replaced with EXOR operator (or any group operator) in Shannon expansion when Post Literals or Generalized Post Literals or Reduced Post Literals as long as they remain disjoint. These expansions can be done for any kind of logic from this chapter. MV Shannon remains thus the most general expansion operator, which is not true for Positive and Negative Davio (pD and nD, respectively), general "orthogonal" (binary and MV) [beyl1]. This is also not true for polynomial and GF expansions in particular. Reverse expansion operations for these expansions exist and are more complicated, which will be presented below. Concluding, the expansions to be considered include generalizations of Shannon and generalizations of Davio, but they have quite different properties. The general method to create a Regular Diagram (of certain type) is as follows. One level of function *f* is expanded (to an assumed type of the Regular Diagram) for a selected variable (or a group of variables). Then, the level of the tree is mapped to the assumed type of Regular

Diagram. This means combining together some nodes of the tree-like lower part of the diagram. Here, however, we have a difference. There is only one combining rule for Shannon reverse expansions but many rules for Davio type (polynomial) expansions. The Regular Diagram expanding procedure requires, in general, repeating some variables in the diagram. The key point was thus to find good methods of variable and expansion types selections. One approach to the variable order and expansion type selection can be based on generalized partial symmetries for cofactors [Jin05, Jeske97]. The examples illustrate that the overhead of variable repeating in planar diagrams and lattices is not excessive.

9.8. Orthogonal Expansion Structures for Quaternary Logic.

Having defined the addition and multiplication in 2^2, we can apply the combinational functions synthesis method based on orthogonal functions presented in [Perkowski93, Perkowski95]. The Figure 9.7.1a shows a block diagram of a structure realizing a function of input variables $x_1, x_2, ..., x_m$. Each column realizes one orthogonal function over 2^2. Multiplied by a constant from 2^2, this function is added to the other orthogonal functions. All operations are in 2^2. The Figure 9.7.1(b) shows an example of realization of one of the functions f_i. Since each cell can realize the identity operation (as shown before), it is possible to omit certain input variables x_i, x_3 in this example. More than one column of cells can be used for the realization of each f_i if necessary. Also, it may be convenient to make certain input variables available on more than one horizontal line. An alternative approach, based on providing literals on horizontal lines, or some functions of single variables which are convenient for the creation of literals, is also possible. In one

such approach, the powers (i.e. multiple products in 2^2) would be used to create polynomial expansions of MVL functions.

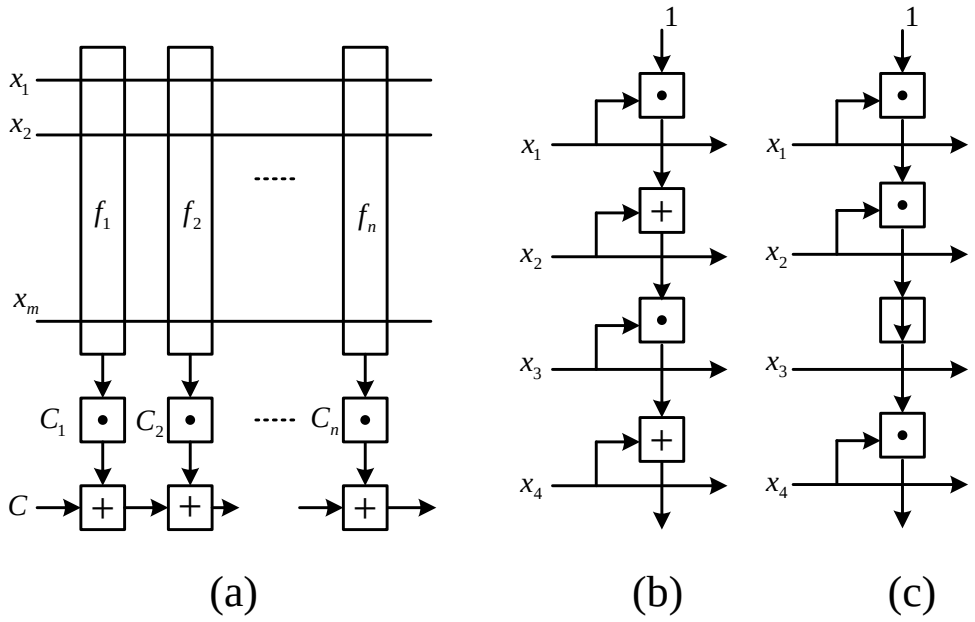


Figure 9.8.1. Example of Generalized PLAs: Orthogonal Expansions in regular PLA-like structures: (a) General structure, (b) orthogonal column with addition and multiplication operators, (c) orthogonal column with multiplication (Galois multiplication, AND, etc) operators.

9.9. Reverse Expansions of Galois Field type.

Reverse expansions for Galois Logic are very similar to the reverse expansions illustrated for Positive and Negative Davio expansions in Chapter 5 and Chapter 6. The formal characterization of all these expansions may be done in line of other expansion families from my dissertation but here I will illustrate only one of such reverse expansions. Please remember that Shannon Reverse Expansions were already presented in Chapter 5 and next chapters.

The Figure 9.9.1 presents the left part of the rule for Reverse Expansion for Davio-0 expansions. The Figure 9.9.2 presents the right part of the rule for Reverse Expansion for Davio-0 expansions. This rule is calculated based on GF(3) algebra. Similarly, the rules for all other Reverse Expansions can be calculated for ternary and quaternary logic, but deriving closed formulas for any radix seems to be very difficult.

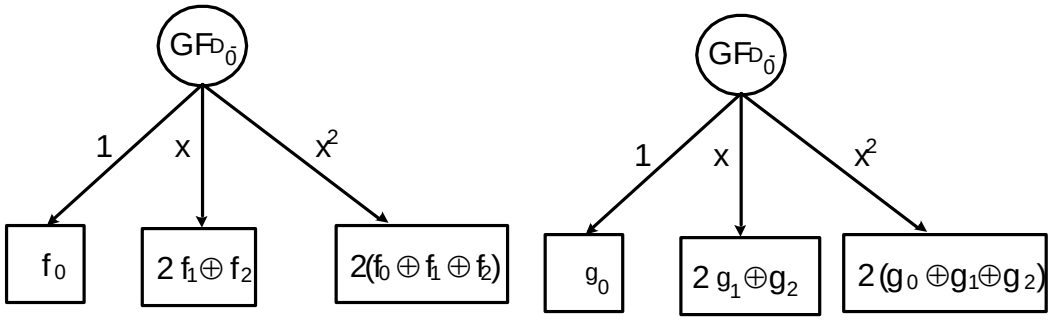


Figure 9.9.1. The left side of the Reverse Expansion for Davio-0 Expansion

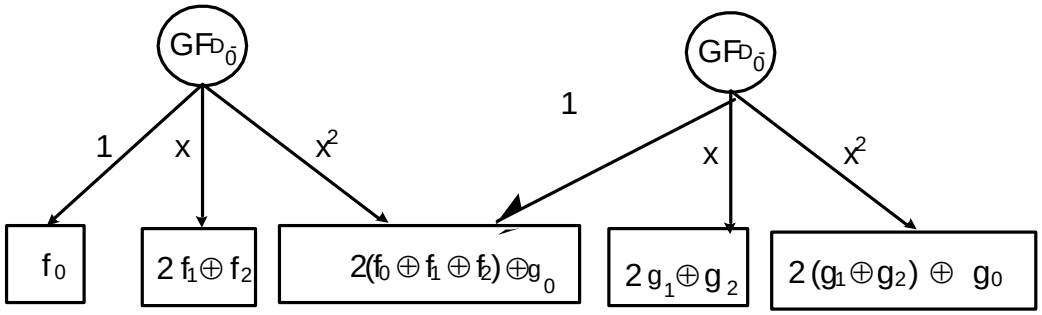


Figure 9.9.2. The right side of the Reverse Expansion for Davio-0 Expansion

The choice of the appropriate regular diagram and array type for a given Boolean function remains a difficult problem to be solved and at the moment we dispose just examples and heuristics, but we do not know any general solution to it. In theory, one would need just the most powerful layout array type, and assume that the design algorithm will select the best expansions and variable ordering in any case (the same as Kronecker Diagram versus BDD or FDD). But creating a good heuristic algorithm for the most general arrays is more difficult than to create such an algorithm for a restricted

type of array. Perhaps, some simpler regular arrays are also better for quantum layout, like assuming only Shannon Expansions (multiplexers) in classical design allows to design the function from mostly pass-transistors and very regular small-grid layout. The Calculation of data input functions to diagram nodes for **any** type of expansions and **any** diagram neighborhoods is performed by the same technique of solving logic equations for a given structure, as one used for Orthogonal logic. This technique is very general and can be adapted to many non-binary logics. However, in contrast to the orthogonal logic, where the equations have always one solution resulting from non-singularity of the matrix M, the structural equations, in general case, can have one, many, or no solutions. (Also, they are no longer equations only over Galois Field.) When there are many solutions, the solution evaluated as the best should be taken. When there are no solutions, the backtrack to another structure, another expansions, or another blocks of input variables should be executed in the synthesizing algorithm.

Selection of the order of (usually repeated) variables is done using the concept of the best separation of most different-value minterms, using repeated variable maps and more general symmetries. The problems of variable ordering and variable partitioning, known for long in logic synthesis to be tough ones, here become even more important, and at the same time more difficult, because the variables must be repeated. Hopefully, it was found that in contrast to the worst-case randomly generated functions, for real-life benchmark functions only few repetitions of variables are enough [Mozammel06, Jeske97]. It is especially easy to **symmetricize** weakly specified functions.

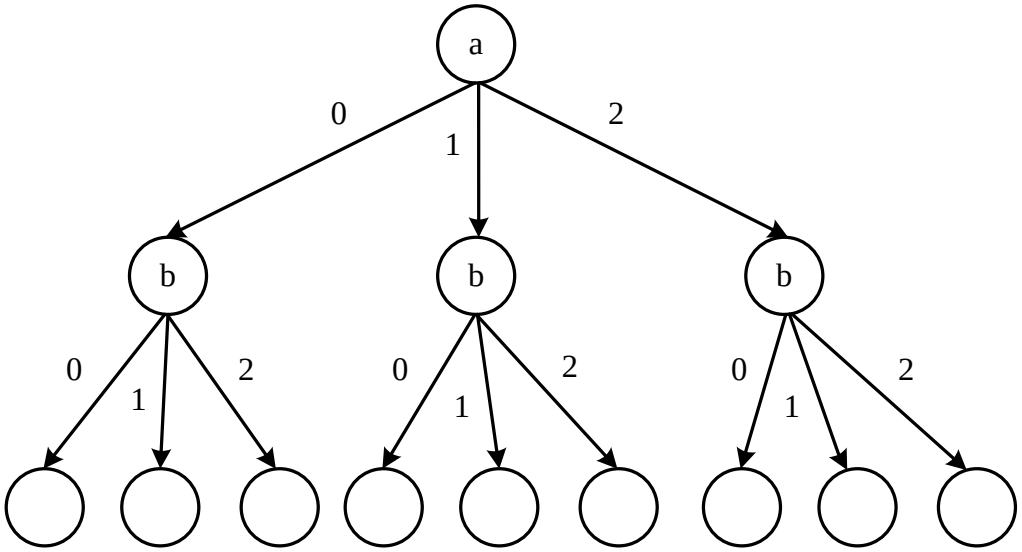


Figure 9.9.3.First three levels of a ternary lattice. The lowest level is for variable c.

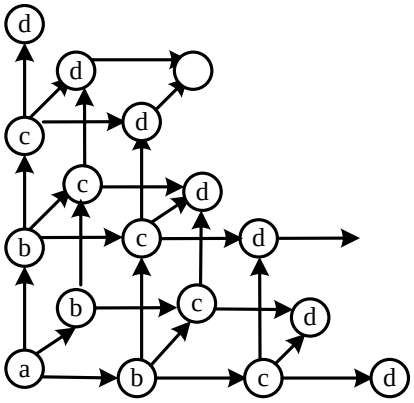


Figure 9.9.4. Regular 3-dimensional lattice for ternary expansions. Every circle shows location of a group of ions in 3D space and letters correspond to ordering of input variables. This diagram illustrates 3-dimensional symmetries of ternary functions.

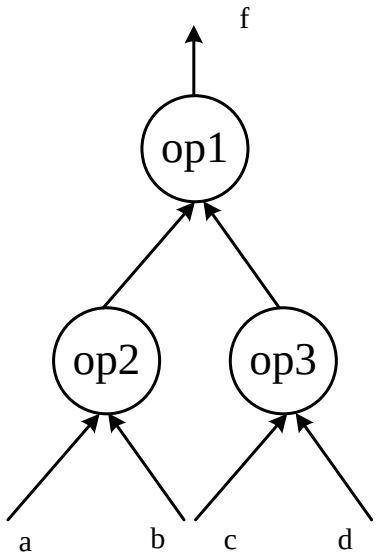


Figure 9.9.5. A small tree of arbitrary non-reversible operators to be mapped.

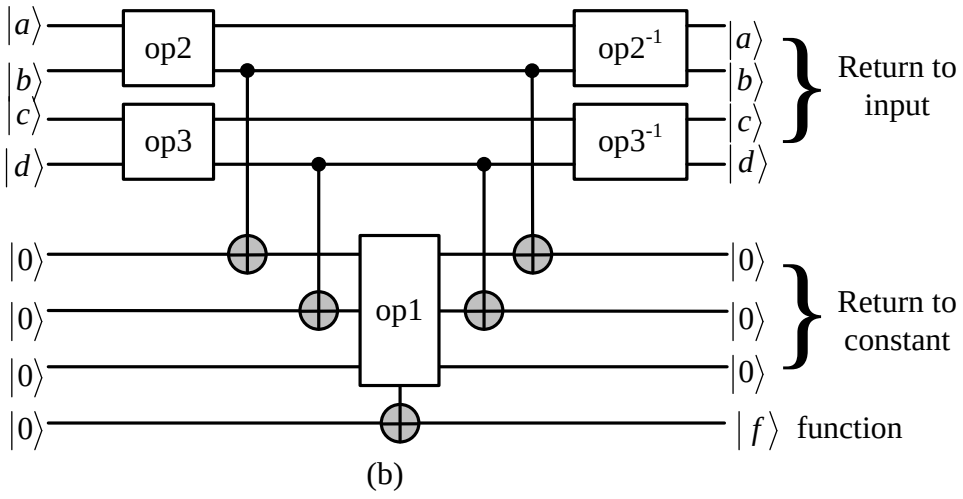


Figure 9.9.6. Mapping to layout with mirrors (a) arbitrary fat tree, (b) its mapping with mirror operators.

Even with non-reversible operators in nodes any trees (including fat trees) can be converted to quantum arrays by adding mirrors. For instance, the tree from Figure 9.9.5 is mapped as in Figure 9.9.6. Similar methods can be applied in 1D, 2D and 3D layouts.

CHAPTER 10

Conclusions and future work.

At the beginning of my research on designing reversible logic circuit, one of the first questions I asked myself was why we need reversible logic circuits? After completing initial research on the subject matter, I was convinced that the logical irreversibility of classical logic gates inadvertently causes the heat dissipation when new values are computed and old information is lost. Landauer [Landauer61] proved that binary logic circuits built using traditional irreversible gates inevitably lead to energy dissipation, regardless of the technology used to realize the gates. Zhirnov *et al.* [Zhirnov03] showed that power dissipation in any future CMOS will lead to an impossible heat removal problem and thus the speeding-up of CMOS devices will be impossible at some point which will be reached before year 2020. Bennett [Bennett73] proved that for power not to be dissipated in a binary logic circuit, it is necessary that the circuit be built from the reversible gates. The operation of a reversible circuit does not lose information and thus not lose energy. Energy is lost only in reading and initialization. *A gate (or circuit) is reversible if it is a one-to-one mapping between sets of input and output values*. Thus all output vectors are just permutations of input vectors. Such a circuit can be described by a binary permutation matrix [Nielsen00]. Bennett's theorem suggests that every future (binary) technology will have to use some kind of reversible gates in order to reduce power dissipation. Concluding, there should be methods to synthesize Boolean circuits using only reversible gates. Several such methods have been developed, but I found on some small examples that all these methods can be improved. This was my main

motivation to develop efficient methodology and software to design and optimize reversible logic circuits.

The main goal of my research is to develop methodology for designing cost effective reversible circuits, using quantum cost model as costs of reversible realization. It means we assume that reversible circuit is realized using quantum gates such as inverters, Toffoli gates and their special cases – Feynman gates (simple 2×2 gates). Quantum costs can be calculated when the circuits are designed with smaller primitives such as quantum gates with two inputs and two outputs. Toffoli gates with many inputs require very many such 2×2 quantum primitives. The benchmark result presented in chapter 6 and chapter 7 demonstrates that quantum cost of reversible circuits increases dramatically with increase in number of qubits operating on the Toffoli gate.

10.1. Key accomplishments of my research.

The key accomplishments of my research are as follows:

- **Development of new algorithm lattice2qa:** I developed a unique method for creating a quantum array from a lattice diagram. Most of the popular algorithms use multi-input (generalized) Toffoli gates for synthesis of reversible logic functions. The methods developed (chapter 6) as a result of my research work use Davio lattice that uses positive Davio gate as a basic building block of the lattice diagram. The logic function realized by positive Davio gate is identical to 3×3 Toffoli gate. Hence the lattice Diagrams (used in classical logic synthesis) can be used for synthesis of reversible logic circuits. With this approach the reversible quantum circuit is realized

using NOT gates, 2×2 Feynman gates and 3×3 (basic) Toffoli gates so costly k×k generalized Toffoli gates (k>> 3) used by all other authors in their algorithms are avoided. The reversible circuits synthesized using my method has following key properties:

1. Circuits are always designed with 3×3 Toffoli gates, Feynman gates, or NOT gates.
2. The algorithm can create a reversible circuit for any arbitrary Boolean function, and not only for reversible functions as it is the case in other algorithms.
3. Creates regular circuits with predictable connections, which is important in some quantum technologies such as linear Ion Trap.

   Some of the drawbacks I observed in the other popular contemporary methods are that they can synthesize only reversible specification of the function. Unfortunately not many real life functions are reversible functions. This means, in order to operate those algorithms, first a given function must be converted into a reversible logic functions. The process of converting non-reversible function to a reversible counterpart inherently adds ancilla bits to the specification of the function. The advantage of my method is that it can synthesize reversible circuit for any given Boolean expression. However it does add small number of ancilla bits that has low cost in some quantum technologies. These ancilla bits are not added arbitrarily but in the process of synthesis. Their number is the same or higher than in the known methods, but the quantum cost is reduced. Results presented in chapters 6 and 7 proved that the overall cost of the circuit is minimized as circuit is built with the smaller primitives.

- **Invention of a new gate (Dipal-gate) and synthesis with layered diagrams:** A new reversible logic gate (Dipal-gate, name given by my advisor Dr. Perkowski) is developed as a result of my research. Dipal-gate is a reversible counterpart of classical multiplexer. This gate was never presented in the literature before. Many variants of the Dipal-gate are created using NOT gates and SWAP gates at input and/or output sides of the Dipal-gate. A quantum array of Dipal gates is generated by synthesizing a Boolean function using a Shannon lattice. The concept of layered diagram is developed as one of the significant contribution of my research. A layered diagram is a synthesis technique that creates layers of blocks consisting of gates from the Dipal-gate family.
- **Development of new synthesis model:** A new synthesis model for 1D and 2D linear ion trap is developed. One of the key advantages of my method is that, the geometry of connections of circuits created by my algorithms is regular and predictable. Every Toffoli gate in the circuit receives one (first) control qubit from the neighboring gate and another (second) control qubit from the variable used in the expansion of the same gate. This makes all connections inside gates and with other gates short. Longer connections are required only to connect Toffoli gates to qubits of input variables. Whereas in other methods any qubit can create quantum gate by interacting with any other qubit located at distance in space. There is a potential technological limitation to realize a quantum gate on qubits at distance. To alleviate this problem of long variable connections in my approach, further I created a new synthesis model in which SWAP gates are inserted on certain places by analyzing the structure of a

lattice diagram such that quantum gates are realized only on the neighboring qubits and long connections are completely avoided. This adds additional cost of SWAP gates inserted automatically in my designs. However this is possibly the best-known method to me to overcome technological limitations caused by distant qubits. When I compared my solutions with those from literature, I have done comparisons twice. First I used the standard quantum cost functions of Maslov [Maslov04] for my method and MMD and Agarwal/Jha methods. This did not require inserting SWAP gates and assumed that gates can be realized on any qubits. My methods proved to be better on many examples. Then I repeated testing my algorithms using the model of Linear Ion Traps in which gates can be realized only on neighbors. To compare "apples with apples and oranges with oranges" I added SWAP gates to solutions created by MMD and Agrawal/Jha algorithms, while the SWAPs were inserted automatically (one SWAP per one Toffoli) by my methods. Again, my solutions were usually better and sometimes much better than those by MMD and Agrawal/Jha approaches. I am not claiming that my methods are always superior to those by other authors. I proved with many examples that for two various cost models my approaches give significant cost reduction for several benchmarks of arbitrary Boolean functions over all existing methods. This is especially true for large functions, with many inputs, outputs and gates.

- **Development of synthesis methods for multiple valued logic:** I presented several layout-driven synthesis approaches and ideas for binary, multi-valued, orthogonal, Galois Field and other types of quantum circuits. In particular, I created lattices, trees, diagrams and regular structures for Multiple-Valued Quantum Circuits. Therefore, the

presented approaches generalize and unify many known expansions, decision diagrams, and regular layout geometries. These methods are of special interest to new technologies such as quantum or quantum dots which require high noise, decoherence and long-line effects in connections. I believe that in future all these effects and phenomena, for one reason or another will cause, the increased interest in regular realizations of binary, multi-valued, and hybrid quantum circuits. The design of individual cells, and specific details of the architecture were determined upon the consideration of the perceived applications of the device. The device is capable of binary and multiple-valued operations. Of course, mathematically, binary signals are a special case of multi-valued logic signals, so everything true for a wider case still remains true for binary. (But 3D layouts cannot be generalized for practical use, it seems). In this dissertation I presented a unified approach to create all MV regular diagrams and regular layouts for MV logic, ternary and quaternary in particular. I was able to present only some of these many examples – as we did not expect at the beginning of my work on this dissertation that the number of such structures is very high. I ended up programming only some of the new methods, but similar programming approaches can be used in future to develop software for other regular structures introduced in this dissertation, as well as for trees and diagrams which are less regular. My methods are also applied to multiple-valued logics extended to reversible circuits and having total or partial regularity. These are new topics that require experimental verification. Ion Trap technology allows ternary and in general multi-valued logic quantum circuits so comparisons of the same functions realized in binary and non-binary quantum circuits will be interesting.

In summary, in this dissertation I presented a complete hierarchy of expansions, trees, diagrams, forms, expressions, and lattices that are related to the regularity of quantum layout. Although it is too early to predict the development and future of quantum technology, if future quantum technologies use multiple-valued circuits then my dissertation will prove to be both pioneering and practical research. Regardless of the progress of technology, the dissertation has a mathematical value – demonstration of new families of regular expansions in two-dimensional and three-dimensional spaces. If for some reasons (for which I see no reasons now) the future quantum circuits will be only binary, the methods from chapters 6 and 7 will be definitely useful. We proved their validity with software and benchmarks.

- **Software development for lattices and creating quantum array:** During the course of my research I developed a software program called **Lattices** to create lattice for any Boolean function. The first version of the program was developed to create positive Davio lattices for positive polarity Reed-Muller functions. In the next stage program was upgraded to create lattices with Shannon cell, which is a basic Dipal gate. I also developed a software program **lattice2qa** to transform lattice into quantum array. The algorithm implemented in **lattice2qa** is one of the key contributions of my research. This algorithm proves the direct link between classical logic synthesis and quantum logic synthesis for the first time. The benchmark results presented in chapter 6 and chapter 7 proved that quantum circuits generated by my software programs are more cost effective compared to other well-known methods.

10.2. Future work.

At many places in this dissertation I have mentioned in context and full detail about the future work especially wherever it was possible to introduce new ideas. In this subsection I will summarize these ideas in very general terms. This dissertation covers quite a broad set of aspects of synthesis to address the problem of designing of reversible logic circuits. The main idea of the dissertation is to create a quantum array from a lattice diagram. In the process of development of this methodology I presented related expansions, trees, diagrams and forms. The methodology developed in this dissertation is robust and convergent for any Boolean function, as lattice diagram can be created for any Boolean function. The software program **Lattices** and **lattice2qa** developed to create lattices and quantum array, have been successfully used on functions up to twenty variables. Vector simulation is performed on the quantum array to validate the final circuit. The Boolean expression for the output function is also derived and validated in some cases. However, there are several topics that need more investigation from the point of view of future research as listed below.

1. To create a quantum array first lattice is created for a given Boolean function and then it is transformed into a quantum array as per the methods presented in chapters 6 and 7. To create the optimum (least number of gates) quantum array, optimization techniques as explained below can be applied during creation of a Lattice diagram as well as by optimizing the quantum array. It is evident that the quantum cost of the array created for a given Boolean function using my algorithm depends on the

number of nodes in the KFDD created for the function. For symmetric functions variable repetition is not required and the number of levels in the KFDD will be same for any order of variables. Also number of levels will be same as number of variables used in the function. However in case of the non-symmetric functions, the joining operation performed to merge neighboring non-isomorphic nodes in the KFDD introduces new variables (that are used for expansion in the previous stage) in the logic functions represented by the merged nodes. This enforces repetition of variables in the subsequent stages. It is hence inevitable to use appropriate ordering of variables to minimize the number of nodes in the KFDD. Following ideas based on the literature can be applied to address variable ordering problem for KFDDs to create optimum KFDDs.

a. The dynamic variable ordering based heuristic approach presented in [Rudell93, Panda94] can be readily used for KFDDs. The Sifting algorithm presented in [Rudell93] for OBDD minimization is based on finding the optimum position for a variable assuming all other variable in a fixed position, the process is repeated for all variables used in the function. However, variable repetition has to be taken into account when using this algorithm for KFDDs.

b. The adjacent isomorphic nodes can be presented with a single node and joining operation is not required. To minimize repetition of variables in KFDDs, selection of variables has to be such that geometric adjacencies are preserved. To increase probability of adjacencies for isomorphic nodes, a flip

Davio operation (for KFDD) presented in Chapter 6 is performed similar to the flip Shannon operation for PSBDD presented in [Wang01].

c. The variable ordering problem for KFDDs can be represented as a permutation problem. However there is one fundamental difference compared to other well-known permutation problems such as "job shop scheduling" and "travelling sales person" (TSP), i.e. the elements of the string (order of variables) need be repeated. This requirement is however supported in ordinary GA. Genetic Algorithm is shown useful to address variable ordering problem for BDDs [Drechsler96, Lenders04]. Similar hybrid GA (HGA) methods can be developed and used for determining optimum variable ordering for KFDDs, with addition of ability to repeat variables in the variable string.

d. The ordering problem can be solved using Simulated Annealing search methods with entropy functions or other evaluations of complexity of Boolean functions. The application of scatter search method presented in [Hung01] for OBDD variable ordering problem produced promising results. Such search methods can be useful for variable ordering problem for KFDDs.

There is much published research on variable ordering for Trees, Decision Diagrams such as BDDs, and even lattices. The variable repetition problems of lattices is of course related to variable ordering, as for some orders the numbers of repeated variables can be decreased. This is a broad research area. In past researchers were writing complete PhDs on the variable ordering problem for BDDs. The same can be done for quantum circuits realized from Lattices.

2. The variable ordering problems should be first solved for lattices that use only Davio expansions. Next they can be extended for all lattices introduced in this dissertation, including Shannon lattices and lattices based on various variants of Dipal gates. First we can do this only for lattices that use the basic Dipal-gate. Subsequently the number of all (generalized) Dipal gates should be calculated and methods of synthesizing lattice with mix of Dipal gates should be developed. Similarly as we have now Pseudo-Kronecker lattices in which Positive Davio, Negative Davio and Shannon cells are mixed in levels and between levels, we can develop concepts and algorithmic tools to synthesize with all possible generalized Dipal gates.
3. The methods presented in this dissertation create Feynman gates and not Toffoli gates on the boundaries of the lattice. The Feynman gates are much cheaper. The variable ordering approach explained earlier in this section as well as transformation of the lattices should be done in such a way that relative ratio of Feynman gates to Toffoli gates is increased.
4. Development of new heuristics methods for utilizing all variants of the Dipal-gate for layered diagrams presented in chapter 7. At this time we can synthesize layered diagrams with basic Dipal-gate, but we do not optimize them.
5. The current approach for creating quantum array from KFDD adds more than minimum number of ancilla bits to the reversible quantum circuit. The numbers of ancilla bits in a quantum array are number of variables used in a function plus number of times variables are repeated. Some of the ancilla bits can be reduced by placing gates on variable bus whenever possible. The template tool can be developed and applied similar to [Dueck03b] for optimizing the quantum array created by our

method. The gates in the quantum array can be swapped to create group of gates to match specific template and replace those gates with compatible smaller number gates.

6. The current methods are for two types of quantum layout: (1) popularly used layout with no constraints on gates location, (2) 1-dimensional layout characteristic for Ion Trap technology. There are several other layouts of ions used or proposed for Ion Traps, for instance two-dimensional, or 3-dimensional. My lattices based synthesis methods can be adapted to these layouts.
7. When all the above listed generalizations and improvements will be done for binary quantum circuits, the gained experience can be next used to develop methods for ternary and other multiple-valued quantum circuits based on concepts and algorithms presented in this dissertation.

Concluding, we can state that the dissertation is only a starting point of several potentially fruitful areas of research related to regularity in synthesis of reversible logic circuits, multiple-valued logic, various quantum technologies (especially similar to Ion Trap), expansions, lattices and trees.

REFERENCES

[A]

- [AgrawalJha04] A. Agrawal and N. K. Jha. "Synthesis of reversible logic," in Proc. *DATE*, Paris, France, pp. 710- 722, February 2004.
- [Adami98] C. Adami, Introduction to Artificial Life, *Springer Verlag*, New York, NY, 1998.
- [Aharonov03] Aharonov D., A. Ta-Shma A.: *Adiabatic Quantum State Generation and Statistical Zero Knowledge.* In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing. ACM Press, New York, 2003, pp. 20–29.
- [Akers59] S.B. Akers, "On a Theory of Boolean Functions," J. of SIAM, vol. 7, pp. 487-498, 1959.
- [Akers72] S.B. Akers, ``A rectangular logic array," IEEE TC, Vol. C-21, pp. 848-857, Aug. 1972.
- [Akers88] S.B. Akers, "On the use of linear assignment algorithm in module placement," 25 Years of Electronic Design Automation, 1988, pp. 218-223.
- [Alcock98] J. Alcock, Animal Behavior: An Evolutionary Approach, *Sinauer Associates*, Sunderland, Massachusetts, 1998.
- [Alexits61] G. Alexits, Convergence Problems of Orthogonal Series, (New York: Pergamon Press, 1961).
- [Allen05] J. Allen, J. Biamonte and M. Perkowski**,** "ATPG for Reversible Circuits using Technology-Related Fault Models," *Proc. International Symposium on Representations and Methodologies for Emergent Computing Technologies*, Tokyo, Japan, September 2005.
- [Almaini89] A. E. A. Almaini., Electronic Logic Systems, second edition (Englewood Cliffs, NJ: Prentice-Hall), Chap. 12, 1989.
- [AlRabadi01] A. Al-Rabadi, and M.A. Perkowski, "Multiple-Valued Galois Field S/D Trees for GFSOP Minimization and Their Complexity". *Proc. ISMVL 2001,* pp.159-166
- [AlRabadi02] A. Al-Rabadi, L. Casperson, M. Perkowski, and X. Song, "Canonical representation for Two-Valued Quantum Computing", Proc. Fifth Intern. Workshop on Boolean Problems, pp. 23-32, September 19-20 2002, Freiberg, Sachsen, Germany.
- [PHDAl-Rabadi] A. Al-Rabadi, "Novel Methods for Reversible Logic Synthesis and Their Application to Quantum Computing", Ph. D. Thesis, PSU, Portland, Oregon, USA, October 24, 2002.
- [AlRabadi04] A.N. Al-Rabadi, "Reversible Logic Synthesis", 2004, Springer, ISBN 3-540-00935-3

- [AlRabadi05] A. N. Al-Rabadi and M. Perkowski, "New Families of Reversible Expansions and their Regular Lattice Circuits," *Journal of Multiple-Valued Logic and Soft Computing (MVLSC),* U.S.A., Volume 11, Number 3-4, 2005.
- [Arabas94] J. Arabas, Z. Michalewicz, and J. Mulawka, "GAVaPS—A genetic algorithm with varying population size," in *Proc. IEEE Int. Conf. on Evolutionary Computation*, Orlando, 1994, pp. 73–78.
- [Arin90] S. Arin, J.L. Deneubourg, S. Goss and J.M. Pasteels, Functional self-Organization Illustrated by Interest Traffic in Ants: The Case of the Argentine Ant, Biological Motion, Lecture Notes in Biomathematics, Edited by W. Alt JOTA: Vol. 115, No. 3, December 2002, 627 and G. Hoffmann, *Springer Verlag*, Berlin, Germany, Vol. 89, pp. 533–547, 1990.

[B]

- [Bae07] J. H. Bae, Ch. B. Bae, G. B. Lee, D. H. Kim, M.A. Perkowski, M.H.A. Khan "Minimization of Ternary and Mixed Binary-Ternary Permutative Quantum Circuits". *Report PSU,* 2007.
- [Baghi83] A. Bagchi, and A. Mahanti, "Search Algorithms udder Different Kinds of Heuristics A Comparative Study," JACM, Vol. 30., 1983, pp. 1-21.
- [Banks71] E. Banks. Information Processing and Transmission in Cellular Automata. MIT PhD. Thesis (1971)
- [Barenco95] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and W. H., "Elementary gates for quantum computation," *The American Physical Society*, no. 5, pp. 3457–3467, 1995.
- [Batisda84] J.R. Batisda, Field Extensions and Galois Theory, (New York: Cambridge University Press, 1984).
- [BrassardGalois03] S. Beauregard, G. Brassard, J. M. Fernandez, Quantum Arithmetic on Galois Fields, *quant-ph/0301163*.
- [Bryant86] {i3} R.E. Bryant, ``Graph-based algorithms for Boolean function manipulation, IEEE TC, Vol. C-35, No. 8, pp. 667-691, 1986.
- [Peres00] H. Bechmann-Pasquinucci and A. Peres, "Quantum Cryptography with 3-State Systems*," Phys. Rev. Lett*. 85, 3313, 2000.
- [Becker95] B. Becker, and N. Gockel, "A Genetic Algorithm for Minimization of Fixed Polarity Reed-Muller Expressions", Proc. of the International Conference on Artificial Neural Networks and Genetic Algorithms, Ales, 1995, pp. 392-395.
- [Bell91] W.J. Bell, Searching Behavior: The Behavioral Ecology of Finding Resources, *Chapman and Hall*, London, England, 1991.

- [Bolling95] B. Bollig, M. Löbbing, and I. Wegener, "Simulated annealing to improve variable orderings for OBDD's," in *Int.Workshop Logic Synthesis*, Granlibakken, CA, May 1995.

- [Bolling96] B. Bollig and I. Wegener, "Improving the variable ordering of OBDD's is NP-complete," *IEEE Trans. Comput.*, vol. 45, pp. 993–1002, Sept. 1996.

- [Bonabeau99] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, *Oxford University Press*, New York, NY, 1999.

- [Bell66] A. W. Bell, Algebraic Structures, (New York: John Wiley & Sons, Inc., 1966).

- [Bennett73] C. H. Bennett, "Logical Reversibility of Computation", *IBM Journal of Research and Development*, 17, 1973, pp. 525-532.

- [Bennett82] C.H. Bennett. The thermodynamics of computation – a review. IJTP, 21(12):905—940, 1982.

- [Bennett89] C.H. Bennett. Time/space trade-offs for reversible computation. SIAM Journal on Computing, pages 766 – 776, 1989.

- [Bennett92] C H Bennett, G Brassard, and N David Mermin. Quantum cryptography without Bell's theorem. *Phys. Rev. Lett.*, 68:557-559, 1992.

- [Beers98] G. E. Beers, K. L. John, "Novel Memory Bus Driver/Receiver Architecture for Higher Throughput", Proc. of the IEEE Int. Conf. On VLSI Design, pp. 259-264, 1998.

- [Bentlet99] Bentlet and J. Bentley, Evolutionary Design by Computers, (San Francisco, California: Morgan Kaufmann Publishers, 1999).

- [Berliner79] H. Berliner, "On the Construction of Evaluation Functions for Large Domains," Proceedings IJCAI-79, Tokyo, Japan, 1979, pp. 53-55.

- [Berry97] J. A. Berry and G. Linoff, Data Mining Techniques: For Marketing, Sales, and Customer Support, (New York: John Wiley & Sons, Inc., 1997).

- [Bertolazzi88] Y. Bertolazzi, and A. Sassano, "A class of polynomially solvable set covering problems," SIAM J. Discrete Math., Vol. L, No. 3., August 1988, pp. 306-316.

- [Bertolazzi87] P. Bertolazzi, arid A. Sassano, "An 0(mm) algorithm for regular set covering problems," Theor. Comput. Sci., Vol. 54., 2,3, Oct. 1987, pp. 237-247.

- [Biamonte04] J. Biamonte, M. Perkowski, Principles of Quantum Fault Diagnostics, McNair research Journal, Issue 1, Volume 1, 2004

- [Biamonte05] J. Biamonte, M. Perkowski, "Automated Test Pattern Generation for Quantum Circuits," McNair Research Journal, Vol. 1, Issue 1, 10 pages, 2005
- [Biamonte05a] J. Biamonte, M. Perkowski, "Tricks to validate quantum switching networks," poster and presentation, Proc. of KIAS-KAIST 6th Workshop on Quantum Information Science, Seoul, Korea, pp. 9, August 22nd - 24th, (2005)
- [Biamonte05b] J. Biamonte, M. Jeong, J. Lee, M. Perkowski, "Extending Classical Test to Quantum," Proceedings of SPIE "Fluctuations and Noise in Photonics and Quantum Optics, Editors: P.R. Hemmer, J.R. Gea-Banacloche, P. Heszler, Sr., M. S. Zubairy, Vol. 5842, pp. 194-205, May (2005), doi: 10.1117/12.623715. III.
- [Biamonte07] J. Biamonte and M/ Perkowski, "A Quantum Test Algorithm," *Submitted to IEEE Transactions on Computers and quant-ph/0501108.*
- [Biamonte05] J. Biamonte, J. Allen, D. Pierce, F. Khan and M. Perkowski, "Automated Test Set Generation for Quantum Circuits," *Proc. International Symposium on Representations and Methodologies for Emergent Computing Technologies,* Tokyo, Japan, September 2005.
- [Biamonte05a] J. Biamonte, J. Allen and M. Perkowski, "Fault Models for Quantum Mechanical Switching Networks," *submitted to Journal on Electronic Testing, 22 pages, (2005), quant-ph/0508147.*
- [Boole54] G. Boole, An Investigation of the Laws of Thought, (London: Walton, 1854). (Reprinted by Dover Books, New York, 1954.)
- [Bourennane01] M. Bourennane, A. Karlsson, and G. Bjoerk, "Quantum key distribution using multilevel encoding," *Phys. Rev. Lett., A,* Vol. 64, 012306/1-5, 2001.
- [Boyer96] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp, "Tight bounds on quantum searching," *Proceedings of PhysComp,* 1996.
- [Bronco95] A. Bronco et al., "Elementary Gates For Quantum Computation", *Physical Review A* 52, 1995, pp. 3457-3
- [Brown90] F.M. Brown, Boolean Reasoning: The Logic of Boolean Equations, (Boston, Massachusetts: Kluwer, 1990)
- [Bruce02] J.W. Bruce, M.A. Thornton, L. Shivakumaraiah, P.S. Kokate, and X. Li, "Efficient Adder Circuits Based on a Conservative Reversible Logic Gate", *Proc. of the IEEE Computer Society Annual Symposium on VLSI,* Pittsburgh, Pennsylvania, April 2002, pp. 83-88.
- [Bruss02] D. Bruss and C. Macchiavello, "Optimal Eavesdropping in Cryptography with Three-Dimensional Quantum States," *Phys. Rev. Lett.* 88, 127901, 2002
- [Brylinski01] J. L. Brylinski, and R. Brylinski, "Universal Quantum Gates," *arXiv: Quant-ph/0108062*

- [Buchanan78] B. G. Buchanan, C.R. Johnson, T. M. Mitchell, arid R. G. Smith, "Models of Learning Systems," in: Belzer, J. (Ed.), Encyclopedia of Computer Science and Technology, 11, Marcel Dekker, New York, 1978, pp. 24-51
- [BullerCellular] A. Buller, M. Perkowski, Cellular Automata of Regular Logic..
- [Bullock05] S.S. Bullock, D.P. O'Leary, and G.K. Brennen, "Asymptotically Optimal Quantum Circuits for d-level Systems, " *Phys. Rev. Lett*. 94, 230502, 2005.
- [Burns98] M. Burns, M. Perkowski, L. Jozwiak, and S. Grygiel, "An Efficient and Effective Approach to Column-Based Input/Output Encoding in Functional Decomposition", Proc. of the 3rd International Workshop on Boolean Problems, Freiberg University of Mining and Technology, Institute of Computer Science, September 17-18, 1998, pp. 19-29.

[C]

- [Carpenter80] B. Carpenter, and IV. N. Davis, "Implementation and performance analysis of parallel assignment algorithms on a hypercube computer," Proc. "Hypercube Concurrent Computers and Applications, Vol. 2., Pasadena, CA, Jan. 19-20, 1980, pp. 1231-1235.
- [Cerf02] N. J. Cerf, M. Bourennane, A. Karlsson and N. Gisin, "Security of Quantum Key Distribution Using d-Level Systems," *Phys. Rev. Lett.* 88, 127902, 2002.
- [Chang98] C. H. Chang and B. J. Falkowski, "Adaptive Exact Optimisation of Minimally Testable FPRM Expansions", IEE Proc. – Computers and Digital Techniques, Nov. 1998, Vol. 145, Issue 6, p. 385
- [Chang99] C.H. Chang, and B.J. Falkowski, " NPN Classification using weight and literal vectors of Reed-Muller expansion" IEEE Electronics Letters, 13th May 1999, Vol. 35 No. 10
- [Cirac95] J I Cirac and P Zoller. Quantum computation with cold, trapped ions. *Phys. Rev. Lett.*, 74(20):4091-4094, May 1995.
- [Cohn62] M. Cohn, "Inconsistent Canonical Forms of Switching functions", IRE Trans. On Electr. Comp., Vol. EC-11, pp. 284-285, 1962.
- [Coon94] B. W. Coon, "Circuit Synthesis through Genetic Programming", Genetic Algorithms at Stanford 1994, Compiled by John R. Koza, (Stanford, California: Stanford University, 1994).
- [Cory97] D.G. Cory, A.F. Fahmy, and T.F. Havel, Nuclear magnetic resonance spectroscopy: an experimentally accessible paradigm for quantum computing, in Proc. of the 4th Workshop on Physics and Computation (Complex Systems Institute, Boston, New England) 1996 Science 275, 350 (1997).

- [Csanky93] L. Csanky, M. Perkowski, I. Schaefer, "Canonical Restricted Mixed-Polarity Exclusive-Or Sums of Products and the Efficient Algorithm for their Minimization," IEE Proceedings, Pt.E, Vol. 140, No. 1, pp. 69 - 77, January 1993.
- [Current95] K. W. Current, "Multiple-Valued Logic Memory Circuit", Int. Journal of Electronics, Vol. 78, No. 3, pp. 547-555, 1995.
- [Curtis04] Curtis, E., Perkowski, M. (2004). A transformation based algorithm for ternary reversible logic synthesis using universally controlled ternary gates. *Proc. IWLS 2004*, Tamecula, California, USA, 2-4 June 2004. pp. 345 – 352.
- [Curtis07] E. Curtis, and M. Perkowski, Minimization of Ternary Reversible Logic Cascades using a Universal Subset of Generalized Ternary Gates, accepted to *International Journal on Multiple-Valued Logic and Soft Computing*, Svetlana Yanushkevich, editor. ISSN 1542-3980. ISI.

[D]

- [Das03] R. Das, A. Mitra, V. Kumar and A. Kumar, "Quantum information processing by NMR: Preparation of pseudo pure states and implementation of unitary operations in a single-qutrit system", *arXiv-quant-ph/0307240v1*, 31 July 2003.
- [Debnath95] D. Debnath and T. Sasao, "GRMIN: A Heuristic Simplification Algorithm for Generalized Reed-Muller Expressions", IFIP WG 10.5, Proc. of the Workshop on Applications of the Reed-Muller Expansion in Circuit Design, 27-29 August 1995, Makuhari, Chiba, Japan.
- [Dotoli01] M. Dotoli, G. Maione, D. Naso, and E. B. Turchiano, "Genetic identification of dynamical systems with static nonlinearities," in *Proc. IEEE SMCia/01,Mountain Workshop Soft Computing Industrial Applications*, Blacksburg, VA, June 25–27, 2001, pp. 65–70.
- [Debnath96] D. Debnath and T. Sasao, "GRMIN2: A Heuristic Simplification Algorithm for Generalized Reed-Muller Expressions, IEE Proc. Comput. Digit. Tech., 143 (6) (1996).
- [Debnath98] D. Debnath, "On the Minimization of AND-EXOR and AND-OR-EXOR Networks", Diss. Kyushu Institute of Technology, Japan, March 1998.
- [DeGaris92] H. de Garis, "Artificial Embryology: The Genetic Programming of an Artificial Embryo", Dynamic Genetic, and Chaotic Programming, Branko Soucek and the IRIS Group, (New York: John Wiley & Sons, Inc. 1992).
- [Denler04] N. Denler, B. Yen, M. Perkowski and P. Kerntopf, "Synthesis of Reversible Circuits from a Subset of Muthukrishnan-Stroud Quantum Realizable Multi-Valued Gates", *Proceedings of IWLS 2004*, Tamecula, California, USA, 2-4 June 2004.

- [Denler04a] N. Denler, B. Yen, M. Perkowski, and P. Kerntopf, ”Minimization of Arbitrary Functions in a New Type of Reversible Cascade built from Quantum-Realizable “Generalized Multi-Valued Gates” , *Proc. IWLS 2004.* pp. 321 – 328.
- [DeVos02] A. De Vos, B. Raa, and L. Storme, “Generating the Group of Reversible Logic Gates”, *Journal of Physics A: Mathematical and General*, vol. 35, 2002, pp. 7063-7078.
- [Dehmelt62] H G Dehmelt. *Phys. Rev.*, 103:1125,1962.
- [Dill97] K. M. Dill and M. A. Perkowski, “Minimization of Generalized Reed-Muller Forms with a Genetic Algorithm”, Proc. of Genetic Programming ’97, July 1997, Stanford University, California.
- [Dill97] K. M. Dill, Growing Digital Circuits: Logic Synthesis and Minimization with Genetic Operators”, M. S. Thesis, Department of Electrical and Computer Engineering, Oregon State University, June 1997.
- [Dill97a] K. M. Dill, K. Ganguly, R. J. Safranek, and M. A. Perkowski, “A New Linearly Independent, Zhegalkin Galois Field Reed-Muller Logic”, Portland State University Department of Electrical and Computer Engineering Report, 1997.
- [Dill97b] K. Dill, J. Herzog, and M. Perkowski, “Genetic Programming and its Application to the Synthesis of Digital Logic”, Proc. of the PACRIM ’97 Conference, Victoria, Canada, Aug. 20-22, 1997, (Piscataway, New Jersey: IEEE 1997).
- [Dill98] K. M. Dill and M. A. Perkowski, “Evolutionary Minimization of Generalized Reed-Muller Forms”, Proc. of the International Conference on Computational Intelligence and Multimedia 1998 (ICCIMA’98), Monash University, Churchill, Vic., Australia, 9-11, February 1998.
- [Dill01] K. Dill, and M. Perkowski, “Baldwinian Learning utilizing Genetic and Heuristic for Logic Synthesis and Minimization of Incompletely specified data with Generalized Reed-Muller (AND-EXOR) forms”, Journal of System Architecture 47, Issue 6, 2001, pp. 477-489.
- [Disman96] M. Disman, “Stalking the Chameleon Computer”, Computer & Communications OEM Magazine, Vol. 3, No. 23, (December/January 1996), pp. 67-73.
- [Drechsler94] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski, ``Efficient representation and manipulation of switching functions based on Ordered Kronecker Functional Decision Diagrams," DAC 1994.
- [Drechsler96] R. Drechsler, B. Becker, and N. Göckel, “A genetic algorithm for variable ordering of OBDD’s,” in *Int. Workshop Logic Synthesis*, Granlibakken, CA, May 1996.

- [Drechsler97] R. Drechsler, "Evolutionary Algorithms for Computer-Aided Design of Integrated Circuits Tutorial", Genetic Programming 1997 Conference, Stanford University, July 13, 1997

- [Drechsler99] R. Drechsler, H. Hengster, H. Schaefer, J. Hartmann, and B. Becker, "Testability of 2-Level AND/EXOR Circuits", Journal of Electronic Testing, Theory and Application, (JETTA), 1999.

- [Drechsler00] R. Drechsler, N. Drechsler, and W. Gunther, "Fast exact minimization of BDD's," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 384–389, Mar. 2000.

- [Drechsler01] R. Drechsler, W. Gunther, and F. Somenzi, "Using lower bounds during dynamic BDD minimization," *IEEE Trans. Computer-Aided Design*,vol. 20, pp. 51–57, Jan. 2001.

- [Dubrova96] E. V. Dubrova and J. C. Muzio, "Testability of Generalized Multiple-Valued Reed-Muller Circuits", Proc. of the 26th International Symposium on Multi-Valued Logic, IEEE, 1996, pp. 56-61.

- [Dubrova97] E. V. Dubrova, "Boolean and Multiple-Valued Functions in Combinational Logic Synthesis", Ph.D. Thesis, University of Victoria, Canada, 1997.

- [Dubrova01] E. Dubrova, Y. Jiang, R. Brayton, "Minimization of Multiple-Valued Functions in Post Algebra", *Proc. IWLS01*, pp. 132-138, June 2001.

- [Dueck03] G. W. Dueck and D. Maslov, "Reversible function synthesis with minimum garbage outputs," in Proc. 6th International Symposium on Representations and Methodology of Future Computing Technologies, Trier, Germany, pp. 154-161, March 2003.

- [Dueck03a] G.W. Dueck and D. Maslov, "Garbage in Reversible Designs of Multiple-Output Functions," Proc. RM 2003, pp. 162 – 170.

- [Dueck86] G. W. Dueck and D. M. Miller, "A 4-Valued PLA Using the MODSUM", Proc. of the 16th International Symposium on Multi-Valued Logic, May 1986, pp. 232-240.

- [Dueck03b] G. W. Dueck, D. Maslov, and D. M. Miller, "Transformation-based synthesis of networks of Toffoli/Fredkin gates," in *Proc. IEEE Canadian Conf. Electrical and Computer Engineering*, May 2003, pp. 211–214.

- [Durt03] T. Durt, N. J. Cerf, N. Gisin and M. Zukowski, "Security of Quantum Key Distribution with Entangled Qutrits," *Phys. Rev. A* 67, 012311, 2003, also *quant-ph/0207057*.

- [Dwave07] http://dwave.wordpress.com/2007/01/19/quantum-computing-demo-announcement/. Look also to many materials linked from this webpage.

- [Dwave07a] DWAVE Corporation: http://dwave.wordpress.com/2007/01/19/quantum-computing-demo-announcement/. Look also to many materials linked from this webpage.

[E]

- [Edward93] H. M. Edward, Galois Theory, (New York: Springer-Verlag, 1993).
- [Einstein35] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?" *Phys. Rev.*, vol. 47, no. 10, pp. 777–780, May 1935.
- [Eberhart95] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. Int. Sym. Micro Machine and Human Science,* Nagoya, Japan, Oct. 1995, pp. 39–43.
- [Eisert99] J. Eisert, M. Wilkens, M. Lewenstein, "Quantum Games and Quantum Strategies" Physical Review Letters 83, 3077 - 3080 1999.
- [Ekert91] A K Ekert. Quantum cryptography based ob Bell's theorem. *Phys. Rev. Lett.*, 67:661-664, 1991.

[F]

- [Fan07] Fan Y.: Generalization of Deutsch-Jozsa algorithm to Multiple-Valued Quantum Logic. Proc. ISMVL 2007, http://ismvl07.ifi.uio.no/.
- [Farhi98] E. Farhi, S.Gutmann, "Quantum computation and decision trees", Phys. Rev. A 58, 915 - 928 (1998).
- [Fei02] X. Fei, D. Jiang-Feng, S. Ming-Jun, Z. Xian-Yi, H. Rong-Dian, and W. Ji-Hui, "Realization of the Fredkin gate by three transition pulses in a nuclear magnetic resonance quantum information processor," *Chinese Phys. Lett.*, vol. 19, no. 8, pp. 1048–1050, 2002.
- [Feynman82] R. Feynman, "Simulating physics with computers", Int. J. Theor. Phys., 21:467, 1982
- [Feynman96] R. Feynman, "Feynman Lectures on Computation", Addison Wesley, 1996
- [Files97] C. Files, R. Drechsler, and M. Perkowski, "Functional Decomposition of MVL Functions Using Multi-Valued Decision Diagrams", Proc. of the International Symposium on Multi-Valued Logic 1997, St. Francis Xavier University, Antigonish, Nova Scotia, Canada, May 28-30, 1997, (Piscataway, New Jersey: IEEE, 1997).
- [Farmer86] J.D. Farmer, N.H. Packard, and A.S. Perelson, The immune system, adaptation, and machine learning, *Physica,* Vol. D, No. 22, 1986, pp.187 – 204.

- [Fleming02] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control system engineering: A survey," *Control Eng. Practice*, vol. 10, pp. 1223–1241, 2002.
- [Fonseca98] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms-Part I: A unified formulation;" "-Part II: Application example,*" IEEE Trans. System, Man, Cybern. A: Systems and Humans*, vol. 28, no. 1, pp. 26–47, Jan. 1998.
- [Files98] C. Files and M. Perkowski, "An Error Reducing Approach to Machine Learning Using Multi-Valued Functional Decomposition", Proc. of the International Symposium on Multi-Valued Logic 1998, Fukuoka, Japan, May 26, 1998, (Piscataway, New Jersey: IEEE, 1998), pp. 167-172.
- [Files98a] C. Files and M. Perkowski, "Multi-Valued Functional Decomposition as a Machine Learning Method", Proc. of the International Symposium on Multi-Valued Logic 1998, Fukuoka, Japan, May 26, 1998, (Piscataway, New Jersey: IEEE, 1998), pp. 173-178.
- [Flitney02] A. P. Flitney, and D. Abbott, "Quantum version of Monty Hall problem,*" Phys. Rev. A*. Vol. 65, 062318, 2002.
- [Fredkin03] E. Fredkin: "An introduction to Digital Philosophy", International Journal of Theoretical Physics, Volume 42, Number 2, pp 189-247 (2003).
- [Fredkin82] E. Fredkin and T. Toffoli, "Conservative logic", *Intern. J. Th. Physics,* 21, pp. 219-253, 1982.
- [Fredkin87] J. Frenk, M. Van Houweninge, and R. Kan, "Order statistics and the linear assignment problem", Computing, N.Y., Vol. 39, April 1, 1987, pp. 165-174.
- [Fujiwara86] H. Fujiwara, "Logic Testing and Design for Testability", Computer Science Series, (Cambridge, Massachusetts: The MIT Press, 1986).

[G]

- [Gamberger97] D. Gamberger and Nada Lavrac, "Conditions for Occam's Razor Applicability and Noise Elimination", Proc. of the 9th European Conference on Machine Learning, Prague, Czech Republic, April 23-25, 1997, (Berlin, Germany: Springer-Verlag, 1997).
- [Gaing04] Z-L. Gaing, "A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System*", IEEE Trans. Energy Con*. Vol. 19, pp. 384-391, No. 2, June, 2004.
- [Glad00] T. Glad and L. Ljung, *Control Theory, Multivariable and Nonlinear Methods*. London, U.K.: *Taylor & Francis*, 2000.

- [Gray98] G. J. Gray, D. J. Murray-Smith, Y. Li, K. C. Sharman, and T. Weinbrenner, "Nonlinear model structure identification using genetic programming," *Contr. Eng. Practice*, no. 6, pp. 1341–1352, 1998.
- [Graunbaum98] D. Grunbaum, Schooling as a Strategy for Taxis in a Noisy Environment, *Evolutionary Ecology*, Vol. 12, pp. 503–522, 1998.
- [Garcia-Ripoll03] J. J. Garcia-Ripoll, P. Zoller, and J. I. Cirac. Speed optimized two-qubit gates with laser coherent control techniques for ion trap quantum computing. *Phys. Rev. Lett.*, 91:157901, 2003.
- [Garfinkel74] R.S. Garfinkel, and G. L. Nerhauser, Integer Programming, Wiley, N. Y., 1972. [s1] A.K. Griffith, "A Comparison and Evaluation of Three Machine Learning Procedures as Applied to the Game of Checkers", Artificial Intelligence, Vol. 5 (1974), pp. 137-148.
- [Gottesman96] D. Gottesman, "A class of quantum error-correcting codes saturating the quantum hamming bound," *phys. Rev. A***54**, p. 1862, 1996.
- [DeGaris92] H. de Garis, "Artificial Embryology: The Genetic Programming of an Artificial Embryo", Dynamic Genetic, and Chaotic Programming, Branko Soucek and the IRIS Group, (New York: John Wiley & Sons, Inc. 1992).
- [DeGaris93] H. de Garis, "Evolvable Hardware: Genetic Programming of a Darwin Machine", Artificial Neural Nets and Genetic Algorithms: Proc. of the International Conference in Innsbruck, Austria, 1993, (New York: Springer-Verlag, 1993).
- [Gershenfeld97] N.A. Gershenfeld and I.L. Chuang, Bulk Spin-Resonance Quantum Computation, Science 275, 350 (1997).
- [Giesecke06] Giesecke N.: *Ternary Quantum Logic*. M.S. thesis, PSU, Dept ECE, 2006.
- [Giesecke07] Giesecke, N., Kim, D. H., Hossain, S., Perkowski, M. (2007). Search for universal ternary quantum gate sets with exact minimum costs. *37th IEEE Int. Symp. On Multiple-Valued Logic (ISMVL 2007)*, Oslo, Norway, 14-15 May 2007. http://ismvl07.ifi.uio.no/
- [Goldberg89] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989.
- [Green91] D. H. Green, "Families of Reed-Muller Canonical Forms", International Journal of Electronics, 70 (1991), pp. 259-280.
- [Greentree04] A. D. Greentree, S. G. Schirmer, F. Green, L. C. L. Hollenberg, A. R. Hamilton and R. C. Clark, "Maximizing the Hilbert Space for a finite Number of Distinquishable States," *Phys. Rev Lett.* 92, 097901, 2004. Also quant-ph/0304050.
- [Grover96] L. Grover, "A fast quantum mechanical algorithm for database search," Proceedings of the 28th Annual ACM Symposium on Theory of

Computing 1996, pp. 212-219 1996. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/9605043

- [Grover97] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79(2):325-328,July 1997.
- [Gruska99] J. Gruska, *Quantum computing*. Osborne/McGraw-Hill,U.S., 1999.
- [Grygiel00] S. Grygiel, *Decomposition of Relations as a new approach to constructive Induction in Machine Learning and Data Mining, Ph.D. dissertation*. Portland State University, 2000.

[H]

- [Hagan96] M. T. Hagan, H. B. Demuth, and M. Beale, Neural Network Design, (New York: PWS Publishing Company, 1996).
- [Hanson93] J. E. Hanson, Computational Mechanics of Cellular Automata. PhD Thesis, Physics Department, University of California, Berkeley, CA, 1993.
- [Hasuo92] Hasuo (Fujitsu Lab Ltd., Japan), "High-Speed Digital Circuits for a Josephson Computer", Proc. of the Int. Symposium on Multi-Valued Logic, pp. 2-8, 1992.
- [Hemmi94] H. Hemmi, J. Mizoguchi, and K. Shimohara, "Development and Evolution of Hardware Behaviors", Artificial Life IV: Proc. of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, Editors: Rodney A. Brooks and Pattie Maes, (Cambridge, Massachussetts: The MIT Press, 1994).
- [Highes00] R.J. Highes, C. P. Williams, "Quantum Computing: The Final Frontier?" IEEE Intelligent Systems, Volume 15, Issue 5 (September 2000), pp. 10-18, ISSN 1541-1672
- [Higuchi93] T. Higuchi, Niwa, Tanaka, Iba, de Garis, and Furuya, "Evolving Hardware with Genetic Learning: A First Step Towards building a Darwin Machine", From Animals to Animats 2: Proc. of the Second International Conference on Simulation of Adaptive Behavior, Editors: Jean-Arcady Meyer, Herbert L. Roitblat, and Stewart W. Wilson, (Cambridge, Massachusetts: The MIT Press, 1993).
- [Higuchi94] T. Higuchi, H. Iba, and B. Manderick, "Evolvable Hardware", Massively Parallel Artificial Intelligence, Chapter 12, Editors: Hiroaki Kitano and James A. Hendler, (Menlo Park, California: AAAI Press / The MIT Press, 1994).
- [Higuchi97] T. Higuchi, Evolvable Hardware Tutorial, Genetic Programming 1997 Conference, Stanford University, July 13, 1997.
- [Higuchi97a] T. Higuchi and M. Iwata, Editors, Evolvable Systems: From Biology to Hardware, (Berlin, Germany: Springer-Verlag, 1997).

- [Hirvensalo01] M. Hirvensalo, " An introduction to quantum computing", Current trends in theoretical computer science: entering the 21st centuary, 2001,World Scientific Publishing Co.,Inc. River Edge, NJ, USA, pp 643-663, ISBN 981-02-4473-8
- [Hochbaum82] D. S. Hochbaum, "Approximation algorithms for the weighted set covering and node covering problems," SIAM J. Comput., Vol. 11, 1982, pp. 535-556.
- [Holland92] J. H. Holland, "Genetic Algorithms", Scientific American, July 1992, pp. 66-72.
- [PHD Sazzad Hossein] S. Hossain, Ph.D. Thesis in preparation, PSU.
- [Hubert87] L. J. Hubert, "Assignment Methods in Combinatorial Data Analysis," Marcel Dekker Inc., New York/Basel, 1987.
- [Hubert84] L. J. Hubert, "Statistical applications of linear assignment," Psychometrica, Vol.49, 1984, pp. 449-473.
- [Hung01] W. Hung, X. Song. BDD Minimization by Scatter Search. IEEE Transactions on Computer-Aided Design, 2001.
- [Hung04] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Provably optimal reversible quantum logic synthesis via symbolic reachability analysis," in *Proceedings of DAC*, 2004.
- [Hung06] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and systems*, vol. 25, no. 9, pp.1652–1663, 2006.
- [Hurst85] S. L. Hurst, D. M. Miller and J. Muzio, "Spectral Techniques in Digital Logic", Academic Press, London, 1985.

[I]

- [lbaraki76] T. lbaraki, "Theoretial comparisons of search strategies in branch-and bound algorithms," Intern. Journal. Comp. Sci., 5, 1976, pp. 315-344.
- [Ilachinski01] A. Ilachinski. Cellular Automata: A Discrete Universe. World Scientific publishing, Singapore, 2001.
- [Iwama02] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits," in Proc. *DAC*, New Orleans, LA, pp. 419-424, June 10-14, 2002.

[J]

- [James87] D. M. James (Space Tech. Corp., USA), "VLSI-MVL Implementation of a Fast Arithmetic Cell with SBNR", IEEE Aerospace Applications Conference Digest, p. 11, 1987.

- [Jin05] Jin S. Zhang, Malgorzata Chrzanowska-Jeske, Alan Mishchenko, Jerry R. Burch, "Fast Computation of Generalized Symmetries in Boolean Functions," *Proceedings of IEEE ASP-DAC, January 2005*.
- [Jeske97] M. Chrzanowska-Jeske, Z. Wang, Y. Xu, "Regular Representation for Mapping to Fine-Grain, Locally- Connected FPGAs," *Proc. Of the International Symposium on Circuits and Systems, ISCAS'97, vol. 4, pp. 2749-2752, 1997.*
- [Jones98] J.A. Jones and M. Mosca, Implementation of a Quantum Algorithm on a Nuclear Magnetic Resonance Quantum Computer, J. Chem. Phys., 109, (1998) 1648
- [Jones98a] J. Jones, R. Hansen and M. Mosca, "Quantum Logic Gates and Nuclear Magnetic Resonance Pulse Sequences," J.Magn.Resonance 135, pages 353-360, (1998), quant-ph/9805070.
- [Juang98] Ch-F. Juang, "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design", *IEEE Trans.* On *Systems, Man and Cybernetics*, Part B, IEEE Trans. Vol. 34, pp. 997-1006, 2, April, 2004.
- [Jonker87] R. Jonker, and Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," Computing, N.Y., Vol. 38., No. 4., March 1987, pp. 325-340.

[K]

- [Kalay98] Kalay, U., Hall, D., Perkowski, M. (1998). A minimal and universal test set for multiple-valued Galois field sum-of-products circuits. *Proc. 7th Workshop on Post-Binary ULSI Systems*, Fukuoka, Japan, May 1998, pp. 50-51.
- [Kalay99] U. Kalay, N. Venkataramaiah, A. Mishchenko, D. Hall, and M. Perkowski, "Highly Testable Finite State Machines Based on Exor Logic", Proc. of the 7th IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing, Victoria, B.C., Canada, August 23-25, 1999.
- [Kalay99a] U. Kalay, M. Perkowski, and D. Hall, "A Minimal Universal Test Set for Self Test of EXOR-Sum-of-Products Circuits", IEEE Transactions on Computers, July 1999.
- [Kalay99b] U. Kalay, M. A. Perkowski, and D. V. Hall, "Highly Testable Boolean Ring Logic Circuits", Proc. of the International Symposium of Multi-Valued Logic 1999, (ISMVL'99).
- [Kalay99c] U. Kalay, D. V. Hall, and M. A. Perkowski, "Easily Testable Multiple-Valued Galois Field Sum-of-Products Circuits", Journal of Multiple-Valued Logic, January 1999.
- [Kari94] J. Kari, Reversibility and surjectivity problems of cellular automata. Journal of Computer and System Sciences, 48(1): pp. 149—182, 1994.

- [Kari96] J. Kari. Representation of reversible cellular automata with block permutations. Mathematical System Theory, 29: pp. 47—61, 1996.
- [Kennedy95] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization," Proc. IEEE Intern. Conf. on Neural Networks, Vol. IV, pp. 1942-1948. IEEE Service Center, Piscataway, NJ, 1995.
- [Kennedy01] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.
- [Kennedy95] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings IEEE International Conferenceon Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.
- [Kempe02] J. Kempe, K. B. Whaley, "Exact gate-sequences for universal quantum computation using the XY-interaction alone," *Phys. Rev. A.* Vol. 65(5), 05230, 2002.
- [Kerntopf04] P. Kerntopf, M. Perkowski and M. H. A. Khan, "On Universality of General Reversible Multiple-Valued Logic Gates" Proceedings of ISMVL 2004, pp. 68-73.
- [Kerntopf04b] P. Kerntopf. "A new heuristic algorithm for reversible logic synthesis," in Proc. *DAC*, pp. 834-837, June 2004.
- [Kerntopf06] P. Kerntopf, M. Perkowski, M. H. A. Khan, Universality of ternary reversible gates. *Accepted to special issue of International Journal on Multiple-Valued Logic and Soft Computing,* Svetlana Yanushkevich, editor. ISSN 1542-3980.
- [Khan03] M. H. A. Khan, M. Perkowski, and P. Kerntopf, "Multi-Output Galois Field Sum of Products Synthesis with New Quantum Cascades," *Proceedings of 3^{3rd} International Symposium on Multiple-Valued Logic, ISMVL 2003,* 16-19 May 2003, Meiji University, Tokyo, Japan, pp. 146-153.
- [Khan04] Khan, M. H. A., Perkowski, M. A., Khan, M. R. (2004). Ternary Galois field expansions for reversible logic and Kronecker decision diagrams for ternary GFSOP minimization. *Proc. of 34th IEEE Int. Symp. on Multiple-Valued Logic (ISMVL 2004)*, Toronto, Canada, 19-22 May 2004, pp. 58-67.
- [Khan04a] M. H. A. Khan and M. A. Perkowski, "Genetic Algorithm Based Synthesis of Multi-Output Ternary Functions Using Quantum Cascade of Generalized Ternary Gates", *Proc. 2004 Congress on Evolutionary Computation*, Portland, OR, USA, 19-23 June 2004, pp. 2194-2201.
- [Khan05] F. Khan, and M. Perkowski, "Decomposition of Ternary Quantum Gates," *Proceedings of RM* 2005.
- [Khan05a] M. H. A. Khan and M. Perkowski, "Quantum Realization of Ternary Parallel Adder/Subtractor with Look-Ahead Carry," *Proc. International*

Symposium on Representations and Methodologies for Emergent Computing Technologies, Tokyo, Japan, September 2005. pp. 15-22.

- [Khan05b] M. H. A. Khan, and M.Perkowski, "Quantum Realization of Ternary Encoder and Decoder," *Proc. International Symposium on Representations and Methodologies for Emergent Computing Technologies*, Tokyo, Japan, September 2005. pp. 23 – 27.
- [Khan05c] S1 M. H. A. Khan, M. Perkowski, M. R. Khan, and P. Kerntopf, Ternary GFSOP minimization using Kronecker decision diagrams and their synthesis with quantum cascades. *Journal of Multiple-Valued Logic and Soft Computing,* ISSN 1542-3980. *11*, 2005, pp. 567-602.
- [Khan05d] M.H.A. Khan and M. Perkowski, Evolutionary Algorithm Based Synthesis of Multi-Output Ternary Functions Using Quantum Cascade of Generalized Ternary Gates, accepted to *special issue of International Journal on Multiple-Valued Logic and Soft Computing,* Tatjana Kalganova, editor. ISSN 1542-3980.
- [Khan05e] Khan M. H. A., Perkowski M.: *Genetic Algorithms Based Synthesis of Multi-Output Ternary Functions Using Quantum Cascade of Generalized Ternary Gates*. special issue of International Journal on Multiple-Valued Logic and Soft Computing, Tatjana Kalganova, editor. In print.
- [Khan06] Khan F., Perkowski M.: *Synthesis of Hybrid and d-Valued Quantum Logic Circuits by Decomposition.* Theoretical Computer Science. Vol. 367, Issue 3, 2006, pp. 336-346.
- [Khan05] M. Khan and M. Perkowski, "Low cost NMR realizations of Ternary and Mixed Quantum Gates and Circuits," *in preparation*, 2005.
- [Khan07] M. Khan, M. Perkowski, D. H. Kim and K. Dill, Investigating Learning Search Strategies for Exploring the Space of Local Equivalence Transformations for Optimization of Quantum Circuits, in preparation.
- [Khlopotine02] A. Khlopotine, M. Perkowski, and P. Kerntopf, "Reversible logic synthesis by gate composition," Proceedings of IWLS 2002, pp. 261 – 266.
- [Kielpinski02] D. Kielpinski, C. Monroe, and D. J. Windeland. Architecture for a large-scale ion-trap quantum computer. *Nature,* 417:709-711, 2002.
- [Kim02] D.H. Kim, Intelligent tuning of a PID controller using a immune algorithm, *Trans. KIEE,* Vol. 51-D, No.1, 2002
- [Kim04] D. H. Kim, "Robust tuning of PID controller with disturbance rejection using bacterial foraging based optimization," *International symposium on computational intelligent and industrial application (ISCIIA2004),* Dec. 20-22, 2004, Hikou, China.
- [Kim04a] D. H. Kim, "Robust PID controller tuning using multiobjective optimization based on clonal selection of immune algorithm," *Proc. Int. Conf.*

Knowledge-based intelligent information and engineering systems, Springer-Verlag. pp. 50-56, 2004.

- [Kim00] J. Kim, J-S. Lee and S. Lee, "Implementing unitary operators in quantum computation, *Physical Review A*, 032312, 2000.
- [Kim06] D. H. Kim, Ch. Brawn, M. Sajkowski, T. Stenzel, T. Sasao, J.Allen, M. Lukac, and M. Perkowski, "Artificial Immune – Fuzzy System to control walking robot Hexor," *submitted to ISMVL 2006.*
- [Kim06a] D. H. Kim, J. I. Park, M. Perkowski, "Intelligent Tuning of a PID Controller Using a Hybrid GA-PSO Approach", *submitted to Conference on Intelligent Control, 2006.*
- [Klimov03] A. B. Klimov, R. Guzman, J. C. Retamal and C. Saavedra, "Qutrit quantum computer with trapped ions," *Phys. Rev. A* 67, 062313, 2003.
- [Knill04] E. Knill, "Fault-tolerant postselected quantum computation: schemes," *quant-ph/0402171*, 2004.
- [Kohavi70] Z. Kohavi, "Switching acid Finite Automata Theory," Mc Gruw-Hill, 1970.
- [KohaviBook] Z. Kohavi, "Switching Circuits and Finite Automata Theory," Mc Graw Hill, 1972.
- [Kooprnans99] T. C. Kooprnans, and M. Beckmann, "Assignment problems and the location of economic activities," Econometrica, 25., pp. 53-76, 1957. [s1] R. M. Karp, "Reducibility Among Combinatorial Problems", Complexity of Computer Computation, ed. Miller, et al., (New York: Plenum), pp. 85-103.
- [Kosko94] B. Kosko, "Fuzzy Systems as Universal Approximators", IEEE Transactions on Computers, Vol. 34, No. 11, 1994, pp. 1329-1333.
- [Koza92] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, (Cambridge, Massachusetts: The MIT Press, 1992).
- [Koza94] J. R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs, (Cambridge, Massachusetts: The MIT Press, 1994).
- [Koza99] J. R. Koza, F. H. Bennett, and D. Andre, Genetic Programming III: Darwinian Invention and Problem Solving, (San Francisco, California: Morgan Kaufmann Publishers, 1999).
- [Kruskal56] J. B. Kruskal, Jr. "On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem", Proceedings of the American Mathematical Society, Vol. 7, No. 1 (Feb., 1956), pp. 48-50.
- [Kristinson92] K. Kristinson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Trans. System, Man, Cybern.*, vol. 22, pp. 1033–1046, Sept.-Oct. 1992.

- [Krohling01] R. A. Krohling and J. P. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms," *IEEE Trans. Evol. Comput*., vol. 5, pp. 78–82, Feb. 2001.
- [Kumar07] M. Kumar, B. Year, N. Metzger, Y. Wang, and M. Perkowski, *Realization of Incompletely Specified Functions in Minimized Reversible Circuits.* Proc. RM 2007. http://ismvl07.ifi.uio.no/
- [Kumar07a] Manjith Kumar's webpage, Available HTTP: http://web.cecs.pdx.edu/~kmanjith/, January 22, 2007

[L]

- [Landauer61] R. Landauer, "Irreversibility and Heat Generation in the Computational Process", *IBM Journal of Research and Development*, 5, 1961, pp. 183-191.
- [Lawler85] E. L. Lawler, J. K. Lenstra, A.H.G. Rinooy Kari, and D.B., Shrnoye, "The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization," Wiley, 1985.
- [Lawrence04] J. Lawrence, "Mutually Unbiased Bases and Trinary Operator Sets for N Qutrits," *arxiv:quant-ph/0403095*.
- [PHDLee] PhD Thesis of Patricia J. Lee. Quantum Information Processing with Two Trapped Camium Ions.
- [Lee06] **S.** Lee, S.J. Lee, T. Kim, J-S. Lee, J. Biamonte, and M. Perkowski, *The Cost of Quantum Gate Primitives,* Journal of Multi-valued Logic and Soft Computing, Vol. 12, No. 5-6. 2006.
- [Lee88] K. F. Lee, and S. Mahajan, "A Pattern Classification Approach to Evaluation FunctionLearning," Artificial Intelligence, Vol. 36., pp. 1-25, 1988.
- [Leibfried03] D. Leibfried, R. Blatt, C. Monroe, D. Wineland, "Quantum dynamics of single trapped ions", Review of Modern Physics, volume 75, 281 (2003).
- [Lenders04] Wolfgang. Lenders, Christel Baier "Genetic Algorithms for the Variable Ordering Problem of Binary Decision Diagrams" in *Int. Workshop Logic Synthesis*, Granlibakken, CA, 2004.
- [Lewandowski94] G. Lewandowski, "Practical Implementation and Applications Of Graph Coloring", PhD thesis, University of Wisconsin-Madison, August 1994.
- [LiFei92] Li-Fei Wu, Marek A. Perkowski, "Minimization of Permuted Reed-Muller Trees for Cellular Logic Programmable Gate Arrays," Proc. of the 2nd Intern. Workshop on Field-Programmable Logic and Applications, FPL'92, Vienna, Austria, pp. 7/4.1-7/4.4, August 31-September 2, 1992.

- [Li06] L. Li, M.A. Thornton, M.A. Perkowski, "A Quantum CAD Accelerator Based on Grover's Algorithm for Finding the Minimum Fixed Polarity Reed-Muller Form," *Proc. ISMVL 2006*, pp. 33.
- [Levy92] S. Levy, Artificial Life: A Report from the Frontier where Computers Meet Biology, *Vintage Books*, New York, NY, 1992.
- [Lin00] Ch-L. Lin, Huai-Wen Su, "Intelligent control theory in guidance and control system design: an Overview," *Proc. Natural. Sci., Counc. ROC(A),* vol. 24, no. 1, pp. 15-30, 2000.
- [Lomont03] Ch. Lomont, "Quantum Circuit Identities," arXiv:quant-ph/0307111v1 16 July 2003.
- [Luger02] G. F. Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, (San Francisco, CA: Addison-Wesley / Pearson Education Limited, 2002).
- [Lukac02] M. Lukac, M. Pivtoraiko, A. Mishchenko, and M. Perkowski, "Automated Synthesis of Generalized Reversible Cascades using Genetic Algorithms", Proc. Fifth Intern. Workshop on Boolean Problems, pp. 33-45, September 19-20 2002, Freiberg, Sachsen, Germany.
- [Lukac02] M. Lukac, and M. Perkowski, "Evolving Quantum Circuits Using Genetic Algorithms", Proc. of 5th NASA/DOD Workshop on Evolvable Hardware 2002, pp. 177- 185.
- [Lukac03] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, Ch-. H. Yu, K.Chung, H. Jee, B. Kim, Y.D. Kim, "Evolutionary Approach to Quantum and Reversible Circuits Synthesis", Artificial Intelligence Review Journal, Special Issue on Artificial Intelligence in Logic Design, S. Yanushkevich guest editor, 2003.
- [Lukac05] M. Lukac, M. Perkowski, "Combining Evolutionary and Exhaustive Search to Find the Least Expensive Quantum Circuits", Proceedings of the 14th International Workshop on Post-Binary ULSI Systems, May 18, 2005, Calgary, Canada
- [Lukac05a] M. Lukac and M. Perkowski, "Using exhaustive search for the discovery of new families of optimum universal permutative binary quantum gates," *Proc. International Workshop on Logic and Synthesis,* June 2005.
- [Lukac07] Lukac M. Lukac and M. Perkowski, "Quantum behaviors: Measurement and synthesis," in Submitted to *Reed-Muller 2007*, 2007.
- [Lukac07a] Lukac M. Lukac, Robots, Emotions, Incompleteness and Quantum Computing, *Ph.D. Thesis in preparation*, PSU, 2007.
- [Lukac07b] M. Lukac and M. Perkowski, "Quantum mechanical model of emotional robot behaviors," in *Proceedings of the ISMVL 2007*, 2007.

- [Laumanns00] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multiobjective evolutionary algorithms with elitism. In *Proc. IEEE Congr. Evol. Comp.*, pages 46–53, Piscataway, NJ, 2000. IEEE Press.

[M]

- [Margolus03] N. Margolus, Universal cellular automata based on the collision of soft spheres. New construction in Cellular Automata, Oxford Press, 2003.
- [Margolus87] N. Margolus, Physics and Computation, MIT PhD Thesis (1987). Reprinted as Tech. Rep. MIT/LCS/TR415, MIT Lab. for Computer Science, Cambridge MA.
- [Maslov03] D. Maslov and G. Dueck, "Improved quantum cost for -bit Toffoli gates," *IEE Electron. Lett.*, vol. 39, no. 25, pp. 1790–1791, Dec. 2003.
- [Maione01] B. Maione, D. Naso, and B. Turchiano, "GARA: A genetic algorithm with resolution adaptation for solving system identification problems," in *Proc. Eur. Control Conf.*, Porto, Portugal, Sept. 4–7, 2001, pp. 3570–3575.
- [Michalewicz99] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. New York: *Springer-Verlag*, 1999
- [Maslov03a] D. Maslov and G. W. Dueck, "Garbage in reversible design of multiple output functions," in *Proc. 6th Int. Symp. Representations and Methodology of Future Computing Technologies*, Mar. 2003, pp. 162–170.
- [Maslov04] D. Maslov and G. W. Dueck. "Reversible cascades with minimal garbage," IEEE Transactions on CAD, 23(11): pp.497-1509, November 2004.
- [Maslov04] D. Maslov, N. Scott, and G. W. Dueck. (2004, Aug.) Reversible logic synthesis benchmarks page. [Online]. Available: http://www.cs.uvic.ca/~dmaslov/
- [Maslov05] D. Maslov, G.W. Dueck, and D.M. Miller, "Synthesis of Fredkin–Toffoli Reversible Networks, IEEE Trans. On Very Large Scale Integration, Vol. 13, No. 6, June 2005, pp. 765-769.
- [Maslov05a] D. Maslov, G.W. Dueck, D.M. Miller, "Tofffoli Network Synthesis with Templates," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2005.
- [Maslov06] D. Maslov, G.W. Dueck, and D.M. Miller, "Fredkin/Toffoli Templates for Reversible Logic,"
- [Maslov05b] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck. "Quantum circuit simplification using templates," in Proc. *DATE*, Munich, Germany, pp. 1208-1213, March 2005
- [McHugh05] D. McHugh and J. Twamley, "Trapped-ion qutrit spin molecule quantum computer," http://arxiv.org/abs/quant-ph/0506031

- [McHugh05a] D. McHugh and J. Twamley, "Quantum Computer Using a Trapped-ion Spin Molecule and Microwave Radiation," *Phys. Rev. A* 71, 012315, 2005.
- [McCluskey97] E. McCluskey and Ch-W. Tseng, Stuck-Fault Tests vs. Actual Defects, 1997
- [Merkle93] R. C. Merkle. "Reversible electronic logic using switches," Nanotechnology, 4:21-40, 1993.
- [McKenzie93] L. McKenzie, A. E. A. Almaini, J. F. Miller, and P. Thompson, "Optimization of Reed-Muller Logic Functions", International Journal of Electronics, 75 (3) (1993), pp. 451-466.
- [MCNC91] MCNC, Benchmark Functions, ftp://mcnc.mcnc.org/, MCNC, 1991
- [Metodi05] Tzvetan S. Metodi, Darshan D. Thaker, Andrew W. Cross, Frederic T. Chong and Isaac L. Chaung. "A Quantum Logic Array Microarchitecture: Scalable Quantum Movement and Computation", International Symposium of Microarchitecture, 2005.
- [Meyer99] D.A. Meyer, "Quantum strategies", Physical Review Letters 82(Feb. 1):1052,1999
- [Michael92] J. L. Michael (Wright Lab, USA), "Heterojunction Bipolar Technology for Emitter-Coupled Multiple-Valued Logic in Gigahertz Adders and Multipliers", Proc. of the International Symposium on Multi-Valued Logic, pp. 18-26, 1992.
- [Michalski86] R. Michalski, J. G. Carbonell, T. M. Mitchell, Machine Learning: An Artificial Intelligence Approach, (Los Altos, California: Morgan Kaufmann, 1986).
- [Milburn00] G J Milburn, S Schneider, and D F V James. Ion trap quantum computing with warm ions. *Fortschr. Phys.*, 48:9-11,801-810, 2000.
- [Miller94] D.M. Miller, "Quantum Circuits Simplification using Templates,"
- [Miller02] D.M. Miller, Spectral and Two-Place Decomposition Techniques in Reversible Logic, IEEE Midwest Symposium on Circuits and Systems, proceedings on CD-ROM, Tulsa, OK, August, 2002.
- [Miller03] D. M. Miller and G.W. Dueck, "Spectral Techniques for Reversible Logic Synthesis," *Proc. RM 2003*, pp. 56-62.
- [Miller03a] D. M. Miller, D. Maslov and G. W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis", Proc. Design Automation Conference, Anaheim, pp. 318–323, June 2003.
- [Miller03] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in Proc. *DAC*, Anaheim, CA, p. 318, June 2-6, 2003.

- [Miller04] D. M. Miller, G. W. Dueck, and D. Maslov, "A Synthesis Method for MVL Reversible Logic", *Proc. 34th Int. Symp. On Multiple-Valued Logic*, Toronto, Canada, 19-22 May 2004, pp. 74-80.
- [Miller04a] D. M. Miller, D. Maslov, and G. W. Dueck, "Synthesis of Quantum Multiple-Valued Circuits", submitted to *Journal of Multiple-Valued Logic and Soft Computing*.
- [Miller05] D. Miller, and E. Fredkin, Two- state, Reversible, Universal Cellular Automata in Three Dimensions. Proceedings of the 2nd Conference on Computing Frontiers, Ischica, Italy, pp. 45—51, 2005.
- [Miller05] D. Maslov, G. W. Dueck, and N. Scott, "Reversible Logic Synthesis Benchmarks," [Online document], Available HTTP: http://www.cs.uvic.ca/~dmaslov/, November 15, 2005 [cited December 5, 2006].
- [Miller06] D. M. Miller, D. Maslov, and G. W. Dueck, "Synthesis of Quantum Multiple-Valued Circuits", *J. MVL.*, Vol. 12, No. 5-6, 2006.
- [Miller94a] J. F. Miller and P. Thomson, "A Highly Efficient Exhaustive Search Algorithm for Optimizing Canonical Reed-Muller Expansions of Boolean Functions", Int. J. of Electron., 76 (1994), pp. 37-56.
- [Miller94b] J. F. Miller, H. Luchian, P.V.G. Bradbeer, and P.J. Barclay, "Using a Genetic Algorithm for Optimizing Fixed Polarity Reed-Muller Expansions of Boolean Functions", Int. J. Electron., 76 (1994), pp. 601-609.
- [Miller97] J. F. Miller, P. Thomson, and T. Fogarty, "Chapter 6 - Designing Electronic Circuits using Evolutionary Algorithms. Arithmetic Circuits: A Case Study", in Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications. Editors: D. Quagliarella, J. Periaux, C. Poloni, and G. Winter. New York: John Wiley & Sons, Inc., 1997.
- [Mintert01] F. Mintert and C. Wunderlich, Ion-Trap Quantum Logic Using Long-Wavelength Radiation, *Phys. Rev. Lett*. 87, 257904, 2001. Also quant-ph/0104041
- [Mirchandaney87] R. Mirchandaney, and J. Stankovic, "Using Stochastic Learning Automata for Job Scheduling in Distributed Processing Systerns," J. Parallel Distrib. Comput., Vol. 3., No. 4., Dec. 1987, pp. 527-552.
- [Mishchenko02] A. Mishchenko and M. Perkowski, "Logic Synthesis of Reversible Wave Cascades", Proc. IEEE/ACM International Workshop on Logic Synthesis, June 4-7 2002, pp. 197 – 202.
- [Mitchell86] T. Mitchell, J. Carbonell, and R. Michalski, (eds.), Machine Learning: A Guide to Current Research: Knowledge Representation, Learning, and Expert Systems, (Nowell, MA: Kluwer Academic Publishers, 1986).

- [Monroe95] C Monroe, D M Meekhof, B E King, W M Itano, and D J Wineland. Demonstration of a fundamental quantum logic gate. *Phys. Rev. Lett.*, 75(25):4714-4717, December 1995.
- [Morita92] K. Morita, S. Ueno, Computation—Universal Models of Two—Dimensional 16-state Reversible Cellular Automata, IEICE Trans. Inf. & Syst , E75-D, 1, pp.141—147, 1992.
- [Morita94] K. Morita. Reversible cellular automata. J. Information Processing Society of Japan, 35: 315—321, 1994.
- [Muthurkrishnan00] A. Muthukrishnan, and C. R. Stroud, Jr., "Multi-valued logic gates for quantum computation", Physical Review A, Vol. 62, No. 5, Nov. 2000, 052309/1-8.

[N]

- [Negotevic02] G. Negotevic, M. Perkowski, M. Lukac, and A. Buller, "Evolving quantum circuits and an FPGA based quantum computing emulator", Proc. Fifth Intern. Workshop on Boolean Problems, September 19-20 2002, Freiberg, Sachsen, Germany, pp. 15-2XX
- [VonNeumann66] J.von Neumann. The Theory of Self-Reproducing Automata. A. W. Burks (ed.), University of Illinois Press, Urbana, IL, 1966.
- [Nielsen97] M. A. Nielsen and Isaac L. Chuang , "Programmable Quantum Gate Arrays", Phys. Rev. Lett. 79, 321 - 324 (1997).
- [Nielsen00] M. Nielsen and I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.
- [Nielsen02] M. A. Nielsen, M. J. Bremner, J. L. Dodd, A. M. Childs and C. M. Dawson, "Universal simulation of Hamiltonian dynamics for quantum systems with finite-dimensional state spaces," *Phys. Rev. A* 66, 022317, 2002
- [Nilsson65] N. J. Nilsson, Learning Machines, McGraw-Hill, New York, 1965.
- [Nilsson71] N.J. Nilsson, "Problem-Solving Methods in Artificial Intelligence," Mc Gram Hill, New York, 1971.
- [Nilsson98] N. J. Nilsson, Artificial Intelligence: A New Synthesis, (San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1998).

[O]

[P]

- [Pakula06] I. Pakula, E.W. Piotrowski, J. Sladkowski, "Quantum market games: implementing tactics measurements" Journal of Physics: Conference Series 30 (2006) pp. 56-59.

- [Paul58] W Paul, O Osberghaus, and E Fischer. *Forschungsber. Wirtsch. Verkehrsminist. Nordrhein-Westfalen*, 415, 1958.
- [Paul90] W. Paul , "Electromagnetic traps for charged and neutral particles", Rev. Mod. Phys, 62, 531,(1990).
- [Passino01] K.M. Passino, Biomimicry of Bacterial Foraging for Distributed Optimization, *University Press*, Princeton, New Jersey, 2001.
- [Passino02] K.M. Passino, Biomimicry of Bacterial Foraging for Distributed Optimization and Control, *IEEE Control Systems Magazine*, 2002.
- [Peres85] A. Peres, "Reversible Logic and Quantum Computers," *Physical Review A*, 32:3266-3276, 1985.
- [Perkowski78] M. Perkowski, "The state-space approach to the design of multipurpose problem-solver for logic design," Proceedings of the IFIP WG.5.2 Working Conference "Artificial Intelligence and Pattern Recognition in Computer-Aided Design", Grenoble, France, 17-19 March 1978, J. C. Latombe (ed.) North Holland, Amsterdam, pp. 124-140, 1978.
- [Perkowski82] M. Perkowski, "Digital Devices Design by Problem-Solving Transformations," Journal on Computers and Artificial Intelligence (Pocitace a umela intelligencia), Vol. 1, No. 4, August 1982, pp. 343-365.
- [Perkowski89] M. Perkowski, M. Helliwell, and P. Wu, "Minimization of Multiple-Valued Input Multi-Output Mixed-Radix Exclusive Sums of Products for Incompletely Specified Boolean Functions", Proc. of the 19th International Symposium on Multiple-Valued Logic, May 1989, pp. 256-263.
- [Perkowski91] M. Perkowski and P. Johnson, "Canonical Multi-Valued Input Reed-Muller Trees and Forms", Proc. of the 3rd NASA Symposium on VLSI Design, University of Idaho, Oct. 30-31, 1991, pp. 11.3.1 – 11.3.13.
- [Perkowski92] M. Perkowski, "Generalized Orthonormal Expansion and Some of Its Applications", Proc. of the International Symposium On Multiple-Valued Logic, Sendai, Japan, May 1992, pp. 442-450.
- [Perkowski92a] M. Perkowski, L. Csanky, A. Sarabi, and I. Schafer, "Minimization of Mixed-Polarity AND/XOR Forms", Proc. of the IEEE International Conference on Computer Design, ICCD'92, October 11-13, 1992, Boston, Massachusetts, pp. 32-36, 1992.
- [Perkowski93] M. A. Perkowski, ``A Fundamental Theorem for EXOR Circuits", Proc. IFIP W.G. 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design, pp. 52--60, Hamburg, Germany, Sep 1993.
- [Perkowski93] M.A. Perkowski, A. Sarabi, and F. R. Beyl, ``Universal XOR Canonical Forms of Switching Functions", Proc. IFIP W.G. 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design, pp. 27--32, Hamburg, Germany, September 1993.

- [Perkowski95] M. Perkowski, A. Sarabi, and F.R. Beyl, "Universal XOR Canonical Forms of Boolean Function and its Subset Family of AND/OR? XOR Canonical Forms", IEEE workshop on Logic Synthesis, 1995.
- [Perkowski95a] M. A. Perkowski, T. Ross, D. Gadd, J. A. Goldman, and N. Song, "Application of ESOP Minimization in Machine Learning and Knowledge Discovery", Proc. of the Second Workshop on Applications of the Reed-Muller Expansion in Circuit Design, Chiba City, Japan, August 27-29, 1995, pp. 102-109.
- [Perkowski97] M. Perkowski, M. Marek-Sadowska, L. Jozwiak, T. Luba, S. Grygiel, M. Nowicka, R. Malvi, Z. Wang, and J. S. Zhang, "Decomposition of Multiple-Valued Relations", Proc. of the International Symposium on Multi-Valued Logic 1997, St. Francis Xavier University, Antigonish, Nova Scotia, Canada, May 28-30, 1997, (Piscataway, New Jersey: IEEE 1997).
- [Perkowski97a] M. Perkowski, L. Jozwiak, and R. Drechsler, "A Canonical AND/EXOR Form that Includes both the Generalized Reed-Muller Forms and Kronecker Reed-Muller Forms", Proc. of the Reed-Muller 1997 Conference, Oxford University, U.K., Sept. 1997, pp. 219-233.
- [Perkowski97b] M. Perkowski, L. Jozwiak, R. Drechsler, and B. Falkowski, "Ordered and Shared, Linearly Independent, Variable-Pair Decision Diagrams", Proc. of the First International Conference on Information, Communications and Signal Processing, ICICS'97, Singapore, September 9-12, 1997, Session 1C1: Spectral Techniques and Decision Diagrams.
- [Perkowski97c] M. Perkowski, L. Jozwiak, and R. Drechsler, ``New Hierarchies of AND/EXOR Trees, Decision Diagrams, Lattice Diagrams, Canonical Forms, and Regular Layouts", Proc. of the Reed-Muller 1997 Conference, Oxford Univ., U.K., Sept. 1997, pp. 115 - 132.
- [Perkowski97d] M. Perkowski, M. Marek-Sadowska, L. Jozwiak, T. Luba, S. Grygiel, M. Nowicka, R. Malvi, Z. Wang, and J. S. Zhang, "Decomposition of Multiple-Valued Relations", Proc. of the International Symposium on Multi-Valued Logic 1997, St. Francis Xavier University, Antigonish, Nova Scotia, Canada, May 28-30, 1997, (Piscataway, New Jersey: IEEE, 1997).
- [Perkowski99] M. Perkowski, R. Malvi, S. Grygiel, M. Burns, and A. Mishchenko, "Graph Coloring Algorithms for Fast Evaluation of Curtis Decomposition", Proc. of the Design Automation Conference, (DAC'99), June 21-23, 1999.
- [Perkowski99a] M. Perkowski, U. Kalay, D. Hall, and A. Shahjahan, "Rectangular Covering Factorization of ESOPs into Scan-Based Levelized Circuits with Universal Test Set", Proc. of Reed-Muller '99, University of Victoria, Victoria, B.C., Canada, August 20-21, 1999.
- [Perkowski99b] M. Perkowski, U. Kalay, D. Hall, and A. Shahjahan, "Rectangular Covering Factorization of ESOPs into Scan-Based Levelized

Circuits with Universal Test Set", Proc. of Reed-Muller '99, University of Victoria, Victoria, B.C., Canada, August 20-21, 1999.

- [Perkowski99c] M. Perkowski, A. Chebotarev, and A. Mishchenko, "Evolvable Hardware or Learning Hardware? Induction of State Machines from Temporal Logic Constraints", Proc. of the First NASA/DOD Workshop on Evolvable Hardware, Jet Propulsion Laboratory, Pasadena, California, July 19-21, 1999.
- [Perkowski99d] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, "Regular realization of symmetric functions using reversible logic," Proceedings of EUROMICRO Symposium on Digital Systems Design, 2001, pp. 245-252.
- [Perkowski99e] M. Perkowski, L. Jozwiak, P. Kerntopf, A. Mishchenko, A. Al-Rabadi, A. Coppola, A. Buller, X. Song, M. M. H. A. Khan, S. Yanushkevich, V. Shmerko, and M. Chrzanowska-Jeske, "A general decomposition for reversible logic," Proceedings of RM 2001. pp. 119 – 138.
- [Perkowski02] M. Perkowski, A. Al-Rabadi, and P. Kerntopf, "Multiple-Valued Quantum Logic Synthesis," Proc. of 2002 International Symposium on New Paradigm VLSI Computing, Sendai, Japan, December 12-14, 2002, pp. 41-47.
- [Perkowski03] M. Perkowski, M. Lukac, M. Pivtoraiko, P. Kerntopf, M. Folgheraiter, D. Lee, H. Kim, W. Hwangboo, J.-W. Kim, and Y.W. Choi, ``A Hierarchical Approach to Computer Aided Design of Quantum Circuits," Proceedings of 6th International Symposium on Representations and Methodology of Future Computing Technology, RM 2003, Trier, Germany, March 10-11, 2003, pp. 201-20X
- [Perkowski04] M. Perkowski, "From Quantum Gates to Quantum Learning: recent research and open problems in quantum circuits", invited paper, Proceedings of 6th International Workshop on Boolean Problems, Freiberg University of Mining and Technology, Germany, September 23-24, 2004, pp. 1-16.
- [Perkowski05] M. Perkowski, "Multiple-Valued Quantum Circuits and Research Challenges for Logic Design and Computational Intelligence Communities," *Invited Paper, IEEE ConneCtIonS,* IEEE Computer Intelligence Society, November 2005, pp. 6-12.
- [Perkowski07] M. Perkowski, J. Biamonte and M. Lukac, "Test Generation and Fault Localization for Quantum Circuits," *Proceedings of the 35th International Symposium on Multiple-Valued Logic,* May 19-May 21, 2005 at Calgary, Canada.
- [Perkowski07a] M. Perkowski, *Quantum Robotics for Teenagers*, book in preparation. 2007.
- [Pierce05] D. Pierce, J. Biamonte and M. Perkowski, "Test Set Generation and Fault Localization Software for Reversible Circuits," *Proc. International*

Symposium on Representations and Methodologies for Emergent Computing Technologies, Tokyo, Japan, September 2005.

- [Pierzchala94] E. Pierzchala, M. A. Perkowski, S. Grygiel, "A Field Programmable Analog Array for Continuous, Fuzzy, and Multi-Valued Logic Applications", Proc. of the International Symposium on Multi-Valued Logic 1994, (ISMVL'94), pp. 148-155, 1994.
- [Pohl87] I. Pohl, "Bi-directorial Search," Proc. of Internatioracal Workshop on Logic Synthesis, Vol. 1+ 2, Research Triangle Park, North Carolina, May 12-15, 1987.
- [Pradhan87] D. K. Pradhan, Fault-Tolerant Computing: Theory and Techniques, Vol. 1, (Upper Saddle River, New Jersey: Prentice-Hall, 1987).
- [Pradhan78] D. K. Pradhan, "Universal Test Sets for Multiple Fault Detection in AND-EXOR Arrays", IEEE Trans. On Comp., Vol. 27, No. 2, pp. 181-187, Feb. 1978.
- [Price99] M. D. Price, S. S. Somaroo, C. H. Tseng, J. C. Core, A.H. Fahmy, T.F. Havel and D. Cory, "Construction and Implementation of NMR Quantum Logic Gates for Two Spin Systems," *Journal of Magnetic Resonance,* 140, pp. 371-378, 1999
- [Price99a] M. D. Price, S. S. Somaroo, A.E. Dunlop, T. F. Havel, and D. G. Cory, "Generalized methods for the development of quantum logic gates for an NMR quantum information processor," *Physical Review A,* Vol. 60, No. 4, October 1999, pp. 2777-2780
- [Pyber86] L. Pyber, "Clique covering of graphs," Combinatorica, Vol. 6., No. 4., 1986, pp. 393-398.

[Q]

- [Quinlan93] J. R. Quinlan, C4.5: Programs for Machine Learning, (Palo Alto, California: Morgan Kaufmann, 1993). [12] J. R. Quinlan, C4.5: Programs for Machine Learning, (Palo Alto, California: Morgan Kaufmann, 1993).

[R]

- [Raussendorf01] R. Raussendorf and H. J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.,* 86:5188-5191, 2001.
- [Raussendorf03] R. Raussendorf, D. E. Browne and H. J. Briegel. Measurement-based quantum computation on cluster states. *Phys. Rev.* A, 68:022312, 2003. quant-ph/0301052.
- [Reddy72] S. M. Reddy, "Easily Testable Realizations for Logic Functions", IEEE Trans. On Comp., Vol. 21 (1972), pp. 1183-1188.

- [Reed54] I.S. Reed. A class of multiple-error-correcting codes and their decoding scheme. *IRE Trans. Of Inf. Theory,* 3:6-12, 1954.
- [Rendell83] L. Rendell, "A New Basis for State-Space Learning Systems and a Successful Implementation," Artificial Intelligence, Vol. 20., 1983, pp. 369-392.
- [Riege92] M. W. Riege, P. W. Besslich, "Low-Complexity Synthesis of Incompletely Specified Multiple-Output Mod-2 Sums", IEE Proc. E., 139 (4) (1992), pp. 355-362.
- [Ross91] T. Ross, M. Noviskey, T. Taylor, D. Gadd, "Pattern Theory: An Engineering Paradigm for Algorithm Design", Report #WL-TR-91-1060, Applications Branch, Mission Avionics Division, Avionics Directorate, Wright Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, July 26, 1991.
- [Risnick94] M. Resnick, Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds, *MIT Press,* Cambridge, Massachusetts, 1994.
- [Rudell85] R. L. Rudell and A. L. Sangiovanni-Vincentelli, "ESPRESSO-MV: Algorithms for Multiple-Valued Logic Minimization", Proc. of the IEEE Custom Integrated Circuits Conference, 1985
- [Rubinstein01] B.I.P. Rubinstein, "Evolving quantum circuits using genetic programming", *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001),* pp. 144-151, 2001.

[S]

- [Sackett00] C A Sackett, D Kielpinski, B E King, C Langer, V Meyer, C J Myatt, M Rowe, Q A Turchette, W M Itano, D J Wineland, and C Monroe. Experimental entanglement of four particles. *Nature,* 404:256-258, March 2000.
- [Salmon89] J. V. Salmon, E. P. Pitty, and M. S. Abramson, "Syntactic Translation and Logic Synthesis in Gatemap", IEE Proceedings, Vol. 136, Part E, No. 4, July 1989, pp. 321-328.
- [Samuel59] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," IBM J., Vol. 3., 1959, pp. 210-129.
- [Samuel67] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers II, Recent Progress", IBM Journal of Research and Development, Vol. 11, (1967), No. 6, pp. 601-617.
- [Sarabi92] A. Sarabi and M. A. Perkowski, "Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/XOR Canonical Networks", Proc. 1992 IEEE Design Automation Conference, June 1992, pp. 30-35.

- [Sarabi93] A. Sarabi and M. Perkowski, “Design for Testability Properties of AND/XOR Networks”, IFIP WG10.5, Proc. of the Workshop on Applications of the Reed-Muller Expansion in Circuit Design 1993, pp. 147-153.
- [Sarabi94] A. Sarabi, N. Song, M. Chrzanowska-Jeske, and M. A. Perkowski, "A Comprehensive Approach to Logic Synthesis and Physical Design for Two-Dimensional Logic Arrays," Proc. DAC 1994.
- [Sarabi99] A. Sarabi, P. F. Ho, K. Iravani, W. R. Daasch, M. A. Perkowski, "Minimal Multi-Level Realization of Switching Functions Based on Kronecker Functional Decision Diagrams," Proc. of IEEE International Workshop on Logic Synthesis, IWLS '93, Tahoe City, CA, pp. P3a-1 - P3a-6, May 1999.
- [Sasao78] T. Sasao, “An Application of Multiple-Valued Logic to a Design of Programmable Logic Arrays”, Proc. of the 8th International Symposium on Multiple-Valued Logic (ISMVL’78), May 1978, pp. 65-72.
- [Stephens86] D. W. Stephens, and J.R. Krebs, Foraging Theory, *Princeton University Press*, Princeton, New Jersey, 1986.
- [Shi98] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *Proc. IEEE World Congress on Computational Intelligence*, May 1998, pp. 69–73
- [Sasao91] T. Sasao, “A Transformation of Multiple-Valued Input To-Valued Output Functions and its Application to Simplification of Exclusive-Or Sum-of-Products Expressions”, Proc. of the International Symposium of Multi-Valued Logic 1991 (ISMVL-91), May 1991, pp. 270-279.
- [Sasao90] T. Sasao and P. Besslich, “On the Complexity of MOD-2 Sum PLAs”, IEEE Transactions on Computers, Vol. 39, No. 2, (February 1990), pp. 262-266.
- [Sasao91a] T. Sasao, “On the Complexity of Some Classes of AND-EXOR Expressions”, IEICE Technical Report FTS 91-35, October 1991.
- [Sasao81] T. Sasao, “Multiple-Valued Decomposition of Generalized Boolean Functions and the Complexity of Programmable Logic Arrays”, IEEE Transactions on Computing, Vol. C-30, September 1981, pp. 635-643.
- [Sasao86] T. Sasao, “MACDAS: Multi-level AND-OR Circuit Synthesis using Two-Variable Generators”, Proc. of the 23rd Design Automation Conference, Las Vegas, June 1986, pp. 86-93.
- [Sasao90a] T. Sasao, “EXMIN: A Simplification Algorithm for Exclusive-Or-Sum-of-Products Expressions for Multiple-Valued Input Two-Valued Output Functions”, Proc. of the International Symposium on Multi-Valued Logic, (ISMVL-90), May 1990, pp. 128-135.
- [Sasao90c] T. Sasao and P. Besslich, “On the Complexity of MOD-2 Sum PLAs”, IEEE Transactions on Computers, Vol. 39, No. 2, (February 1990), pp. 262-266.
- [Sasao91d] T. Sasao, “A Transformation of Multiple-Valued Input To-Valued Output Functions and its Application to Simplification of Exclusive-Or Sum-of-

Products Expressions", Proc. of the International Symposium of Multi-Valued Logic 1991 (ISMVL-91), May 1991, pp. 270-279.

- [Sasao91e] T. Sasao, "On the Complexity of Some Classes of AND-EXOR Expressions", IEICE Technical Report FTS 91-35, October 1991.
- [SasaoBook1992]
- [Saul92] J. Saul, "Logic Synthesis for Arithmetic Circuits using the Reed-Muller Representation", Proc. of the European Conf. on Design Automation 1992, pp. 109-113.
- [Sasao93e] T. Sasao, Logic Synthesis and Optimization, (Norwell, Massachusetts: Kluwer Academic Publishers, 1993)
- [Sasao94] T. Sasao and D. Debnath, "An Exact Minimization Algorithm for Generalized Reed-Muller Expressions", Proc. of the IEEE Asia-Pacific Conference on Circuits and Systems, (APCCAS'94), Taipei, Taiwan, pp. 460-465, Dec. 1994.
- [Sasao95f] T. Sasao and M. Perkowski, EXOR Logic Synthesis (Boston: Kluwer Academic Publishers, 1995), pp. 19-32.
- [Sasao95g] T. Sasao, "Easily Testable Realizations for Generalized Reed-Muller Expressions", IEEE Trans. Comp., Vol. 46, No. 6, pp. 709-716, 1997.
- [Schaefer91] I. Schaefer and M. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms", Proc. of the International Symposium on Multi-Valued Logic, (ISMVL'91), May 1991, pp. 40-48.
- [Schaefer93] I. Schaefer and M. Perkowski, "Synthesis of Multi-Level Multiplexer Circuits fo Incompletely Specified Multi-Output Boolean Functions with Mapping Multiplexer Based FPGAs", IEEE Transactions on Computer Aided Design, Vol. 12, No. 11, November 1993, pp. 1655-1664.
- [Schnidt-Kaler03] F Schmidt- Kaler, H Haffner, M Riebe, S Gulde, G P T Lancaster, T Deuschle, C Becher, C F Roos, J Eschner, and R Blatt. Realization of the circ-zoller controlled-not quantum gate. *Nature*, 422:408-411, 2003.
- [Schrödinger26] E. Schrödinger, "Quantisierung als Eigenwertproblem", Annalen der Physik 79, 361(1926).
- [Shannon49] C. E. Shannon, "The Synthesis of Two-Terminal Switching Circuits", Bell System Technical Journal, Vol. 28, pp. 59-98, January 1949.
- [Shende02] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Reversible logic circuit synthesis", in Proc. Int. Conf. Computer-Aided Design, San Jose, CA, pp. 125-132, November 10-14, 2002.
- [Shende02a] V.V. Shende, A.K. Prasad, I.L. Markov, J.P. Hayes, "Reversible Logic Circuit Synthesis," *Proc. 11th IEEE/ACM Intern. Workshop on Logic Synthesis (IWLS),* 2002, pp. 125 – 130.

- [Shende03] *V.V. Shende, A.K. Prasad, I.L. Markov and J.P. Hayes, Synthesis of Reversible Logic Circuits, IEEE Trans. on CAD 22, 710 (2003).*
- [Shende03a] *V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," IEEE Trans. Computer-Aided Design Integr. Syst., vol. 22, no. 6, pp. 723–729, Jun. 2003.*
- [Shende04] Shende et al new 2004 and 2005 matrix decomposition papers.
- [Shivgand05] V.S. Shivgand, A.Aulakh, and M. Perkowski, "Quantum Circuit Layout," *Proc. International Symposium on Representations and Methodologies for Emergent Computing Technologies,* Tokyo, Japan, September 2005.
- [Shor94] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring" in Proc. 35th Annu. Symp. On the Foundations of Computer Science (ed. Goldwasser, S.) 124-134 (IEEE Computer Society Press, Los Alamitos, California, 1994).
- [Shor97] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26, 1484-1509 (1997).
- [Sipper97] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer, "A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems", IEEE Transactions on Evolutionary Computation, April 1997, Vol. 1, No. 1, pp. 83-97.
- [Sleator94] T. Sleator and H. Weinfurter, "Realizable universal quantum logic gates," preprint 1994.
- [Smolin96] J. Smolin, D. P. DiVincenzo, "Five two-qubit gates are sufficient to implement the quantum Fredkin gate." Physical Review A, Vol. 53, no. 4, April 1996, pp. 2855-2856.
- [Song05] X. Song, G. Yang, and M. Perkowski, ``Algebraic Characteristics of Reversible Gates," Accepted to *Theory of Computing Systems (Mathematical Systems Theory),* Springer Verlag. First published on Online Test, ISSN 1432-4350. 2005.
- [Slagle70] J. It. Slagle, "Artificial Intelligence: the Heuristic Programming Approach," McGraw Hill, New York 1970.
- [Software1] Software may be run at PSU from the directory: /stash/polo/benchmarks/MCNC.
- [Song93] N. Song, "Minimization of Exclusive Sum of Products Expressions for Multiple-Valued Input Incompletely Specified Functions", Master of Science Thesis, Portland State University, 1993.
- [Spector99] L. Spector, H. Barnum, H.J. Bernstein, and N. Swamy, "Finding a better-than-classical quantum AND/OR algorithm using genetic programming*,"*

Proc. 1999 Congress on Evolutionary Computation, Vol. 3, pp. 2239-2246, Washington DC, 6-9 July 1999, IEEE, Piscataway, NJ.

- [Spekkens02] R. W. Spekkens and T. Rudolph, "Degrees of concealment and bindingness in quantum bit commitment protocols," *Phys. Rev. A* 65, 012310, 2002.
- [Stankovic97] R. Stankovic and R. Drechsler, "Circuit Design from Kronecker Galois Field Decision Diagrams for Multiple-Valued Functions", Proc. of the International Symposium on Multi-Valued Logic 1997, St. Francis Xavier University, Antigonish, Nova Scotia, Canada, May 28-30, 1997, (Piscataway, New Jersey: IEEE, 1997).
- [Steane97] A. Steane, "The ion trap quantum information processor", Appl. Phys. B. 64, 623 (1997).
- [Stedman05] Ch. Stedman, B. Yen and M. Perkowski, "Synthesis of Reversible Circuits with Small Ancilla Bits for Large Irreversible Incompletely Specified Multi-Output Boolean Functions" *Proceedings of the 14th International Workshop on Post-Binary ULSI Systems*, May 18, 2005, Calgary, Canada.
- [Stewart89] I. Stewart, Galois Theory, 2nd Edition, (New York: Chapman & Hall, 1989).
- [Styer02] D. F. Styer *et al.*, "Nine formulations of quantum mechanics", American Journal of Physics 70, 288 (2002).
- [Svore04] K. Svore, T.G. Draper, S.A. Kutin, and E.M. Rains, "Logarithmic-depth Quantum Carry-lookahead Adder, *quant-ph/04061 42*

[T]

- [Tanaka96] U Tanaka, H Imajo, K Hayasaka, R Ohmukai, M Watanabe, and S Urabe. Determination of the ground-state hyperfine splitting of trapped 113cd+ ions. *Phys. Rev. A*, 53:3982-3985, 1996.
- [Terhal99] B. M. Terhal, I.L. Chuang, D.P. DiVincenzo, M. Grassl and J.A. Smolin, "Simulating quantum operations with mixed environments," *quant-ph/9806095*, *Phys. Rev A.* 60, pp. 881-885, 1999.
- [Thew02] R. T. Thew, K. Nemoto, A.G. White, and W.J. Munro, "Qudit Quantum State Tomography,*" Phys. Rev. A*. Vol. 66, 012303, 2002.
- [Thompson03] A. Thompson, I. Harvey, and P. Husbands, "Unconstrained Evolution and Hard Consequences", School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK (Electronically available from: http://www.cogs.susx.ad.uk/users/adrianth.).
- [Thompson95] A. Thompson, "Evolving Electronic Robot Controllers that Exploit Hardware Resources", Advances in Artificial Life: Proc. of the Third European Conference on Artificial Life, Granada, Spain, June 4-6, 1995, F.

Moran, A. Moreno, J. J. Merelo, and P. Chacon, Editors, (Berlin, Germany: Springer-Verlag, 1995.)

- [Thompson95a] A. Thompson, "Evolving Fault Tolerant Systems", First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, (GALESIA'95), Sheffield, September.
- [Toffoli90] T. Toffoli and N. Margolus. Invertible cellular automata: A review. Physica D, 45:229—253, 1990.
- [Tsai96] C. Tsai and M. Marek-Sadowska, "Multilevel Logic Synthesis for Arithmetic Functions", Proc. of the 33rd Design Automation Conference (DAC), Las Vegas, NV, 1996.
- [Tanese89] R. Tanese, "Distributed genetic algorithm," in *Proc. Int. Conf. Genetic Algorithms*, 1989, pp. 434–439.
- [Tsutsui02] S. Tsutsui and D. E. Goldberg, "Simplex crossover and linkage identification: Single-stage evolution vs. multi-stage evolution," in *Proc. IEEE Int. Conf. Evolutionary Computation*, HI, 2002, pp. 974–979.

[U]

[V]

- [Varma91] D. Varma and E. A. Trachtenberg, "Computation of Reed-Muller Expansions of Incompletely Specified Boolean Functions from Reduced Representations", IEE Proc. E., 138 (2) (1991), pp. 85-92.
- [Vasko88] F. Vasko, and F. Wolf, "Solving large set covering problems on a personal computer," Comput. Oper. Res., Vol. 15., No. 2., Febr. 1988, pp. 115-121.
- [Vedral96] V. Vedral, A. Barenco, and A. Ekert, "Quantum Networks for elementary arithmetic operations," *Phys. Rev. A.* 54:147, 1996

[W]

- [Wang01] W. Wang, M. Chrzanowska-Jeske, "A Global Approach to the Variable Ordering Problem in PSBDDs, " *Proc. of the IEEE International Symposium on Circuits and Systems, ISCAS*, 2001.
- [Williams99] C.W. Williams, A.G. Gray, "Automated Design of Quantum Circuits", ETC Quantum Computing and Quantum Communication, *QCQC '98*, Palm Springs, California, February 17-20, *Springer-Verlag*, pp. 113-125, 1999.
- [Winter74] D. J. Winter, The Structure of Fields, (Berlin, Germany: Springer-Verlag, 1974).
- [Wineland79] D J Wineland and W M Itano. Laser cooling of atoms. *Phys. Rev. A*, 20(4):1521-1540, October 1979.

- [Windeland83] D J Wineland, W M Itano and R S VanDyck. High resolution spectroscopy of stored ions. *Adv. Atom. Mol. Phys.*, 19:135-186, 1983.
- [Wineland98] D. J. Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, and D. M. Meekhof, "Experimental Issues in Coherent Quantum-State Manipulation of Trapped Atomic Ions", Journal of Research of the National Institute of Standards and Technology 103, 259 (1998).
- [Wireworld01] http://mathworld.wolfram.com/WireWorld.html
- [Wolfram02] S. Wolfram. A new kind of Science. Wolfram Media, 2002.
- [Wu96] H. Wu, M. Perkowski, X. Zeng, and N. Zhuang, "Generalized Partially-Mixed-Polarity Reed-Muller Expansion and Its Fast Computation", IEEE Transactions on Computers, Vol. 45, No. 9, September 1996, pp. 1084-1088.

[X]

[Y]

- [Yang04] G. Yang, W. Hung, X. Song, and M. Perkowski, "Exact synthesis of 3-qubit quantum circuits from non-binary quantum gates using multiple-valued logic and group theory," *International Journal of Electronics,* 2004.
- [Yang04a] G. Yang, W. Hung, X. Song, and M. Perkowski, "Depth Limited Realization of Reversible Logic," *Microelectronics,* March 2004.
- [Yang04b] G. Yang, X. Song, and M. Perkowski, "Minimal Universal Library," *Discrete Applied Mathematics,* 2004.
- [Yang05] G. Yang, W. N. N. Hung, X. Song and M. Perkowski, "Exact Synthesis of 3-qubit Quantum Circuits from Non-binary Quantum Gates Using Multiple-Valued Logic", IEEE/ACM Design Automation and Test in Europe (DATE), Munich, Germany, March 2005.
- [Yang05a] G. Yang, X. Song, W. N. N. Hung, and M. Perkowski, "On Realization of 3-qubit Reversible Circuits with the minimum number of non-linear gates", 7th International Symposium on Representations and Methodology of Future Computing Technologies (RM2005), Tokyo, Japan, September 2005.
- [Yang05b] G. Yang, W. Hung, X. Song, and M. Perkowski, "Majority-Based Reversible Logic Gates", *Theoretical Computer Science* C. 334(1-3), pp. 259-274, April 15, 2005. ISSN 0304-3975.
- [Yang05b] G. Yang, X. Song, M. Perkowski, Fast Synthesis of Exact Minimal Reversible Circuits using Group Theory, *ACM/IEEE ASP-DAC (Asia and South Pacific Design Automation Conference),* Shanghai, People's Republic of China, January 2005.

- [Yang05c] G. Yang, X. Song, M. Perkowski, "Bi-direction synthesis for reversible circuits," *Proc. IEEE Computer Society Annual Symposium on VLSI*, 2005.
- [Yang05d] G. Yang, X. Song, M. Perkowski, and W.N.N. Hung, "Minimal Universal Library for n*n Reversible Circuits," *Proc. International Symposium on Representations and Methodologies for Emergent Computing Technologies,* Tokyo, Japan, September 2005.
- [Yang05e] G. Yang, X. Song, M. Perkowski, and J. Wu, "Realizing ternary quantum switching networks without ancilla bits," *Journal of Physics A. Mathematical and General*, 2005.
- [Yang05f] G. Yang, X. Song, W. Hung and M. Perkowski, "Bi-directional synthesis for reversible circuits," *IEEE Transactions on VLSI Systems*, 2005
- [Yang05g] G. Yang, X. Song, and M. Perkowski, "On realization of 3-qubit reversible circuits," *Journal of Circuits, Systems, and Computers (JCSC)*, World Scientific Publishers, 2005.
- [Yang05h] G. Yang, X. Song, M.A. Perkowski, W.N.N. Hung, and J.Biamonte, "The Power of Large Pulse-Optimized Quantum Libraries: Every 3-qubit Reversible Function can be Realized with at Most Four Levels," *Proc. International Workshop on Logic and Synthesis*, June 2005.
- [Yang06] G. Yang, F. Xie, X. Song and M.A. Perkowski, "Universality of two-qudit ternary reversible gates", *J. Phys. A: Math. Gen.,* 2006, Vol. **39,** pp. 7763-7773.
- [Yen05] B. Yen, P. Tomson, and M. Perkowski, **"**Sum of Non-disjoint Cubes Covering Generation for Multi-Valued Systems of base 2, for use in Muthukrishnan-Stroud Quantum Realizable Gates: An Extension of the EXOR Covering Problem", *Proceedings of* IWLS 2005. June 2005.
- [Yoshida00] H. Yoshida, K. Kawata, and Y. Fukuyama, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment*", IEEE Trans. Power Syst.,* Vol. 15, pp. 1232-1239, Nov. 2000.
- [Yu88] C. Yu, and B. Walt, "Learning dominance relations in combinatorial search problems," JEEP Trans. Software Engng., Vol. 14., No. 8., August 1988, pp. 1155-1175.

[Z]

- [Zakrevskij95] A. Zakrevskij, "Minimum Polynomial implementation of Systems of Incompletely Specified Boolean Functions", IFIP WG 10.5, Proc. of the Workshop on Applications of the Reed-Muller Expansion in Circuit Design, August 27-29, 1995, Makuhari, Chiba, Japan.
- [Zalka99] Ch. Zalka, "Grover's Quantum Searching algorithm is optimal", *Phys. Rev. A* 60, pp. 2746–2751, 1999.

- [Zeng95] X. Zeng, M. Perkowski, K. Dill, and A. Sarabi, "Approximate Minimization of Generalized Reed-Muller Forms", IFIP WG 10.5, Proceedings of the Workshop on Applications of Reed-Muller Expansion in Circuit Design, 27-29 August 1995, Makuhari, Chiba, Japan, pp. 221-230.
- [Zhang99] W. Zhang, State Space Search. Algorithms, Complexity, Extensions, and Applications, Springer, 1999.
- [Zhegalkin29] I. L. Zhegalkin, "Arithmetization of Symbolic Logic", (in Russian), Mathematiceskij Sbornik, Vol. 35, pp. 311-373, (1928), Vol. 36, pp. 205-338 (1929).
- [Zhirnov03] V. V. Zhirnov, R. K. Kavin, J. A. Hutchby, and G. I. Bourianoff, "Limits to Binary Logic Switch Scaling – A Gedanken Model", *Proc. of the IEEE*, 91, no. 11, 2003, pp. 1934-1939.
- [Zilic93] Z. Zilic, Z. G. Vranesic, "Multiple-Valued Logic in FPGAs", Proc. of the Midwest Symposium on Circuits and Systems, Vol. 2, pp. 1553-1556, 1993.
- [Zilic95] Z. Zilic and Z. Vranesic, "A Multiple-Valued Reed-Muller Transform for Incompletely Specified Functions", IEEE Trans. On Comput., 44 (8) (1995), pp. 1012-1020.
- [Zilic93a] Z. Zilic, Z. G. Vranesic, "Current-mode CMOS Galois Field circuits", Proc. of the International Symposium on Multi-Valued Logic 1993, (ISMVL'93), pp. 245-250, 1993.
- [Zilic02] Z. Zilic and K. Radecka, "The Role of Super-Fast Orthogonal Transforms in Speeding up Quantum Computations," ISMVL 2002.
- [Zupan97] B. Zupan, M. Bohanec, I. Bratko, and J. Demˇsar, "Machine learning by function decomposition," in *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 421–429. [Online]. Available: citeseer.ist.psu.edu/article/zupan97machine.html